US 20090141806A1

(19) United States

(12) Patent Application Publication

MARPE et al.

(10) Pub. No.: US 2009/0141806 A1

(43) Pub. Date: Jun. 4, 2009

(54) VIDEO FRAME ENCODING AND DECODING

(75) Inventors: **Detlev MARPE**, Berlin (DE); **Heiko Schwarz**, Berlin (DE); **Thomas Wiegand**, Berlin (DE)

Correspondence Address:
GLENN PATENT GROUP
3475 EDISON WAY, SUITE L
MENLO PARK, CA 94025 (US)

(73) Assignee: **Fraunhofer-Gesellschaft zur Foerderung der angewandten Forschung e.V.**, Munich (DE)

(21) Appl. No.: **12/324,792**

(22) Filed: **Nov. 26, 2008**

Related U.S. Application Data

(62) Division of application No. 10/769,403, filed on Jan. 30, 2004.

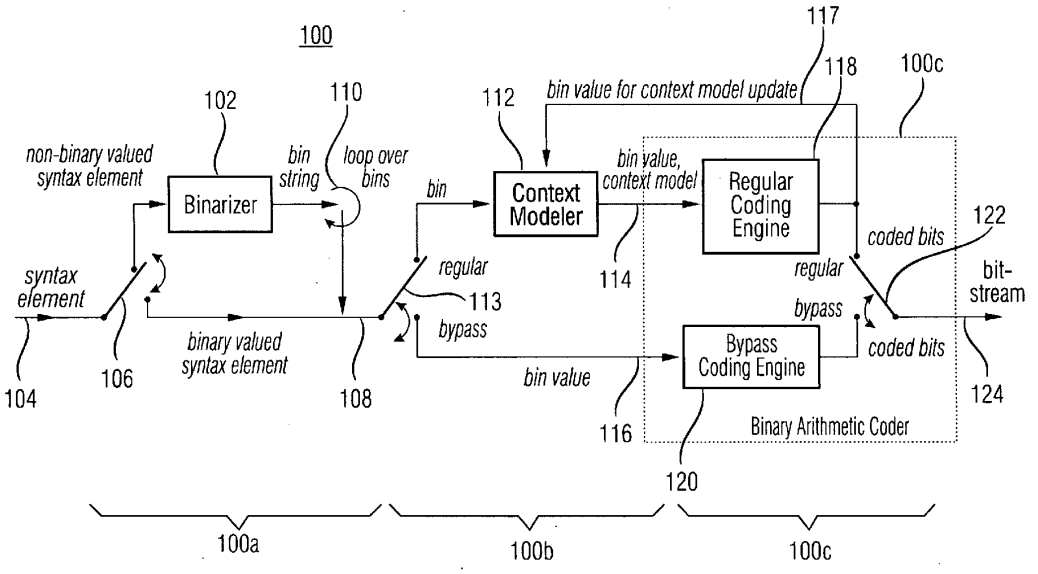
Publication Classification

(51) **Int. Cl.**
H04N 7/26 (2006.01)

(52) **U.S. Cl.** **375/240.24**; 375/E07.2

(57) **ABSTRACT**

A video frame arithmetical context adaptive encoding and decoding scheme is presented which is based on the finding, that, for sake of a better definition of neighborhood between blocks of picture samples, i.e. the neighboring block which the syntax element to be coded or decoded relates to and the current block based on the attribute of which the assignment of a context model is conducted, and when the neighboring block lies beyond the borders or circumference of the current macroblock containing the current block, it is important to make the determination of the macroblock containing the neighboring block dependent upon as to whether the current macroblock pair region containing the current block is of a first or a second distribution type, i.e., frame or field coded.



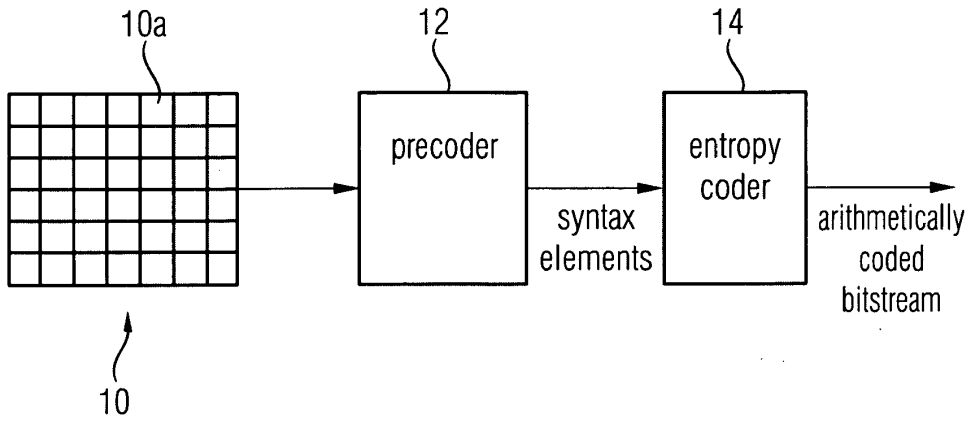

10a
12
14
precoder
entropy coder
syntax elements
arithmetically coded bitstream
10

FIG 1

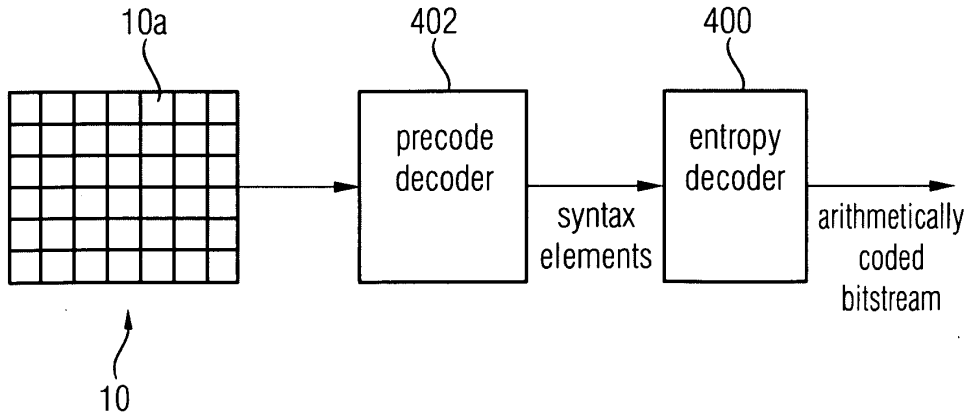

10a
402
400
precode decoder
entropy decoder
syntax elements
arithmetically coded bitstream
10

FIG 10

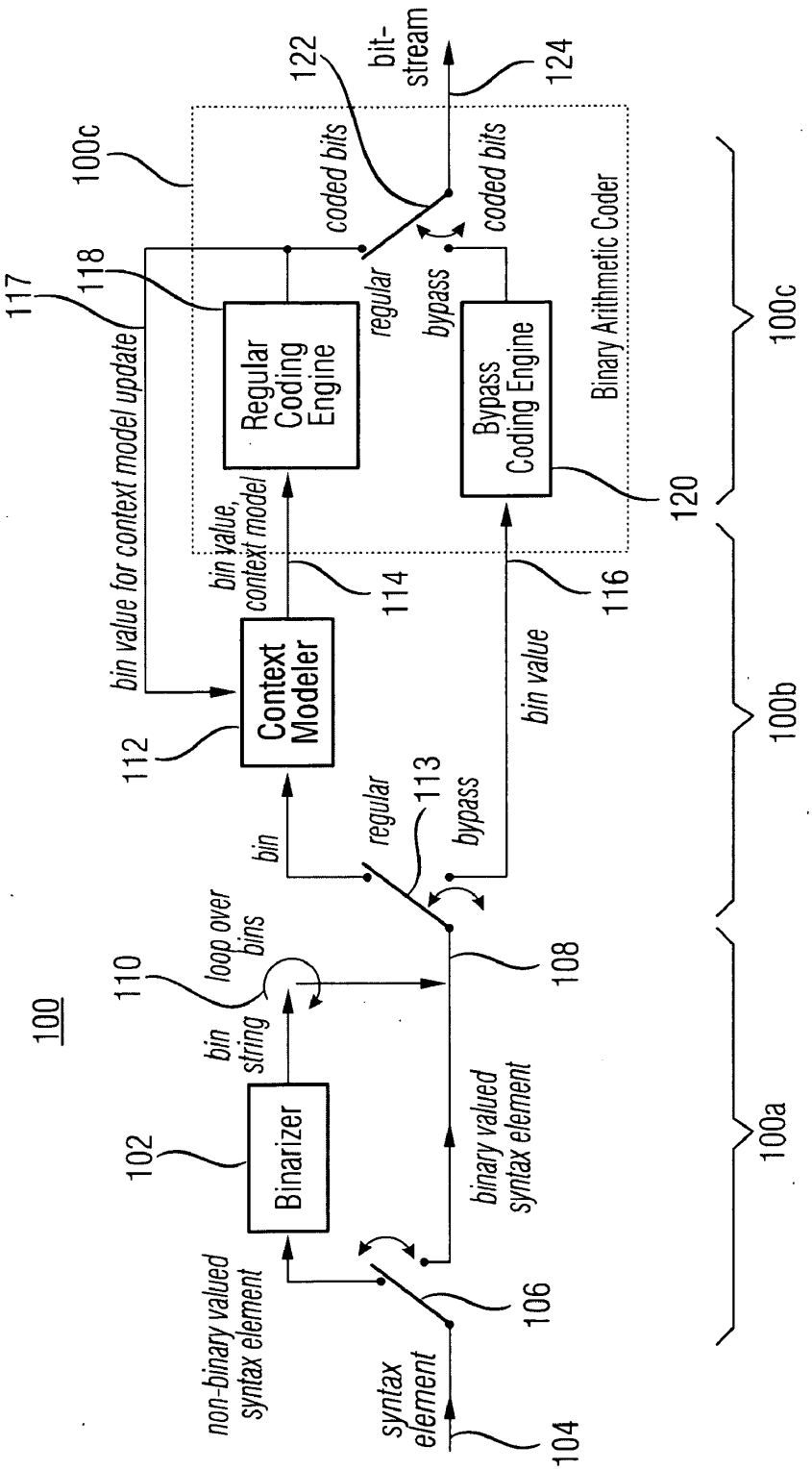

100
102
Binarizer
bin string
110
loop over bins
non-binary valued syntax element
syntax element
104
106
binary valued syntax element
108
bin
regular
113
bypass
112
Context Modeler
bin value for context model update
117
118
bin value, context model
114
Regular Coding Engine
bin value
116
Bypass Coding Engine
120
Binary Arithmetic Coder
100c
regular
bypass
coded bits
122
coded bits
bit-stream
124
100a
100b
100c

FIG 2

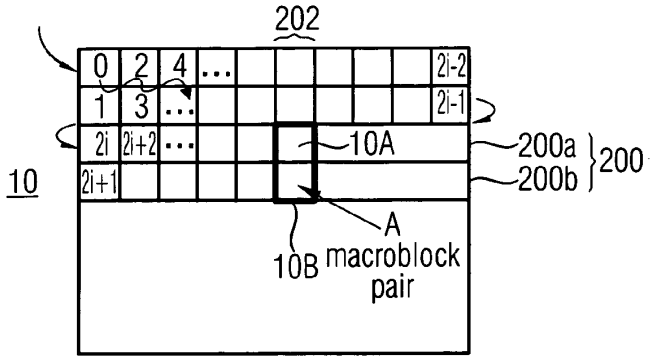

202
0
2
4
...
2i-2
1
3
...
2i-1
2i
2i+2
...
10A
200a
200
200b
10
2i+1
A
10B
macroblock
pair

FIG 3

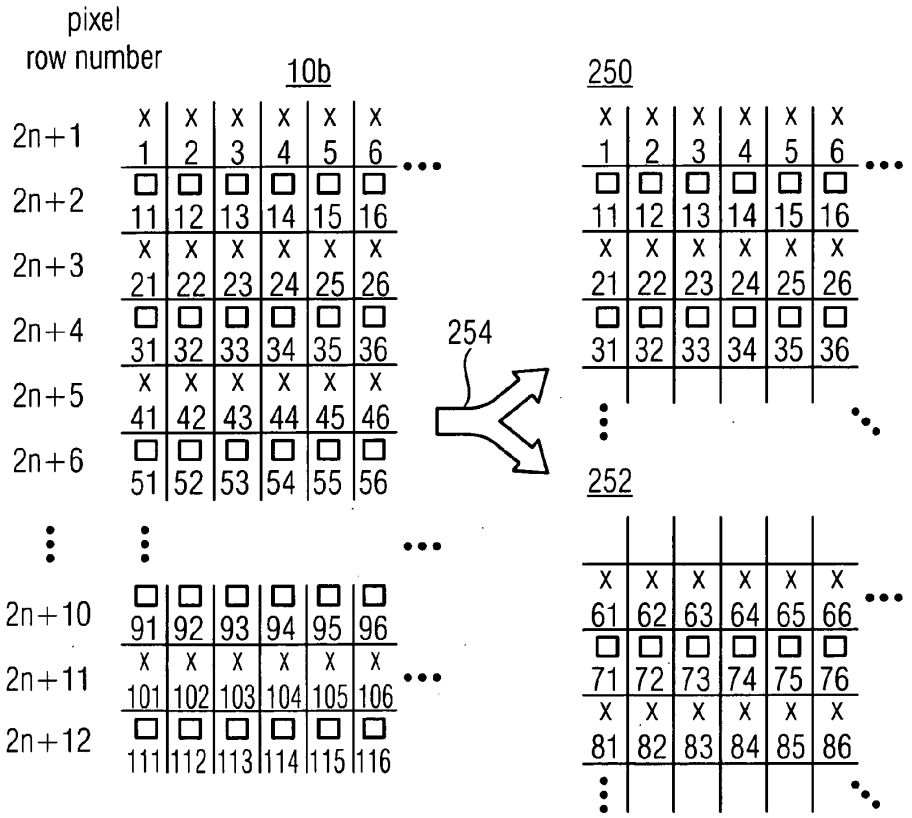

pixel
row number
10b
250
2n+1
2n+2
2n+3
2n+4
2n+5
2n+6
2n+10
2n+11
2n+12
254
252

FIG 4A

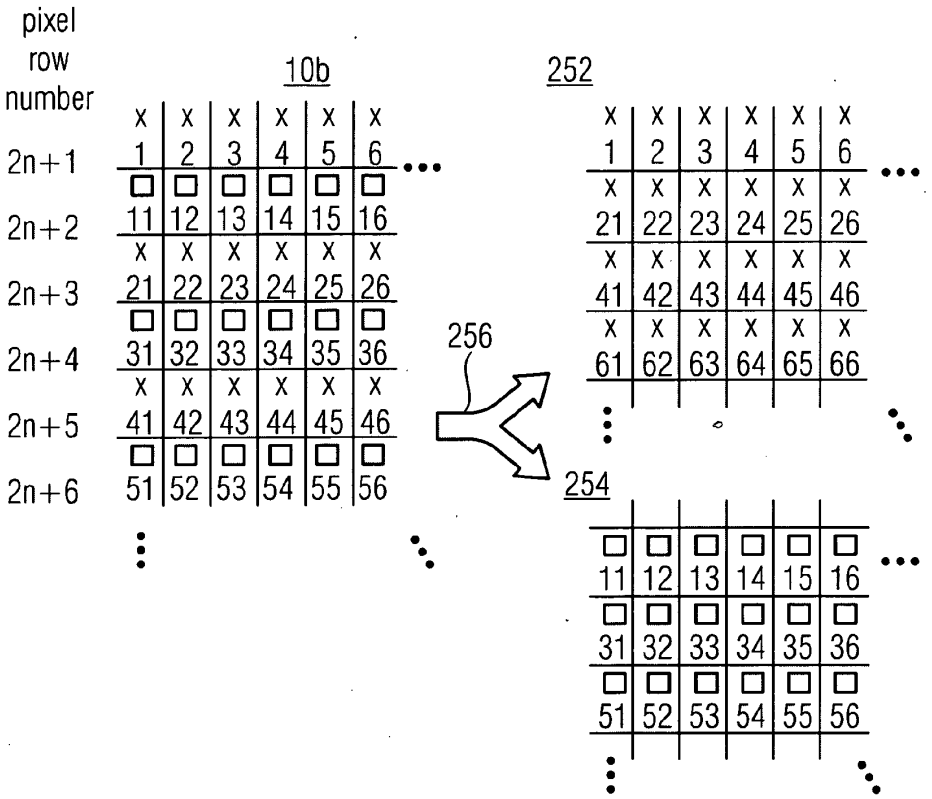

pixel row number
10b
252
256
254
2n+1
2n+2
2n+3
2n+4
2n+5
2n+6

FIG 4B

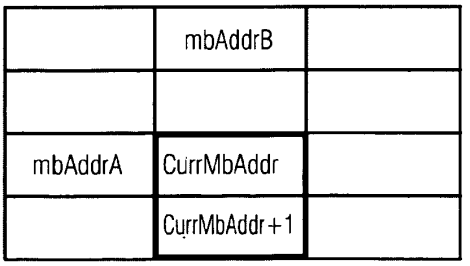

mbAddrB
mbAddrA
CurrMbAddr
CurrMbAddr+1

FIG 7

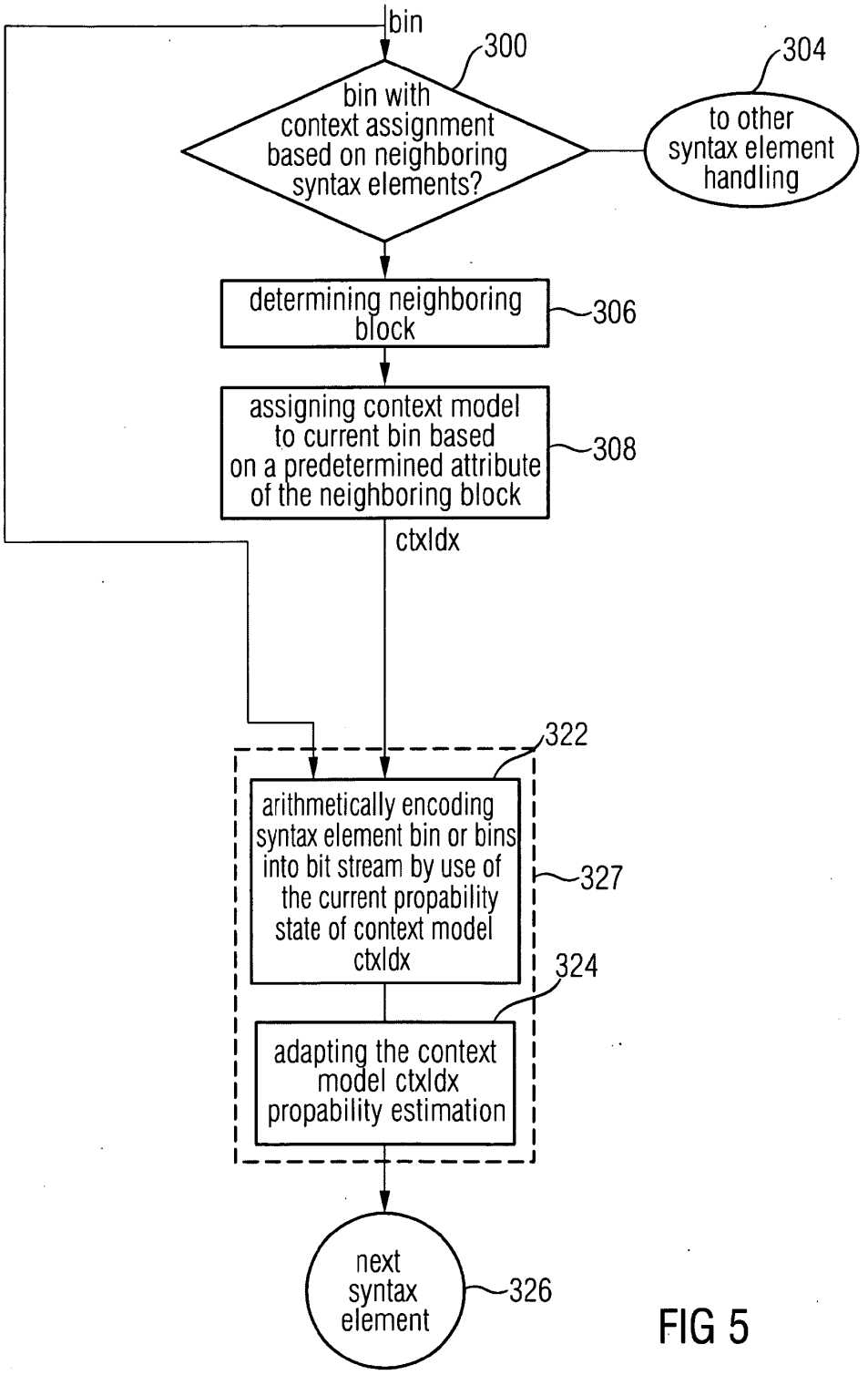

bin
300
bin with context assignment based on neighboring syntax elements?
304
to other syntax element handling
determining neighboring block
306
assigning context model to current bin based on a predetermined attribute of the neighboring block
308
ctxIdx
322
arithmetically encoding syntax element bin or bins into bit stream by use of the current propability state of context model ctxIdx
327
324
adapting the context model ctxIdx propability estimation
next syntax element
326

FIG 5

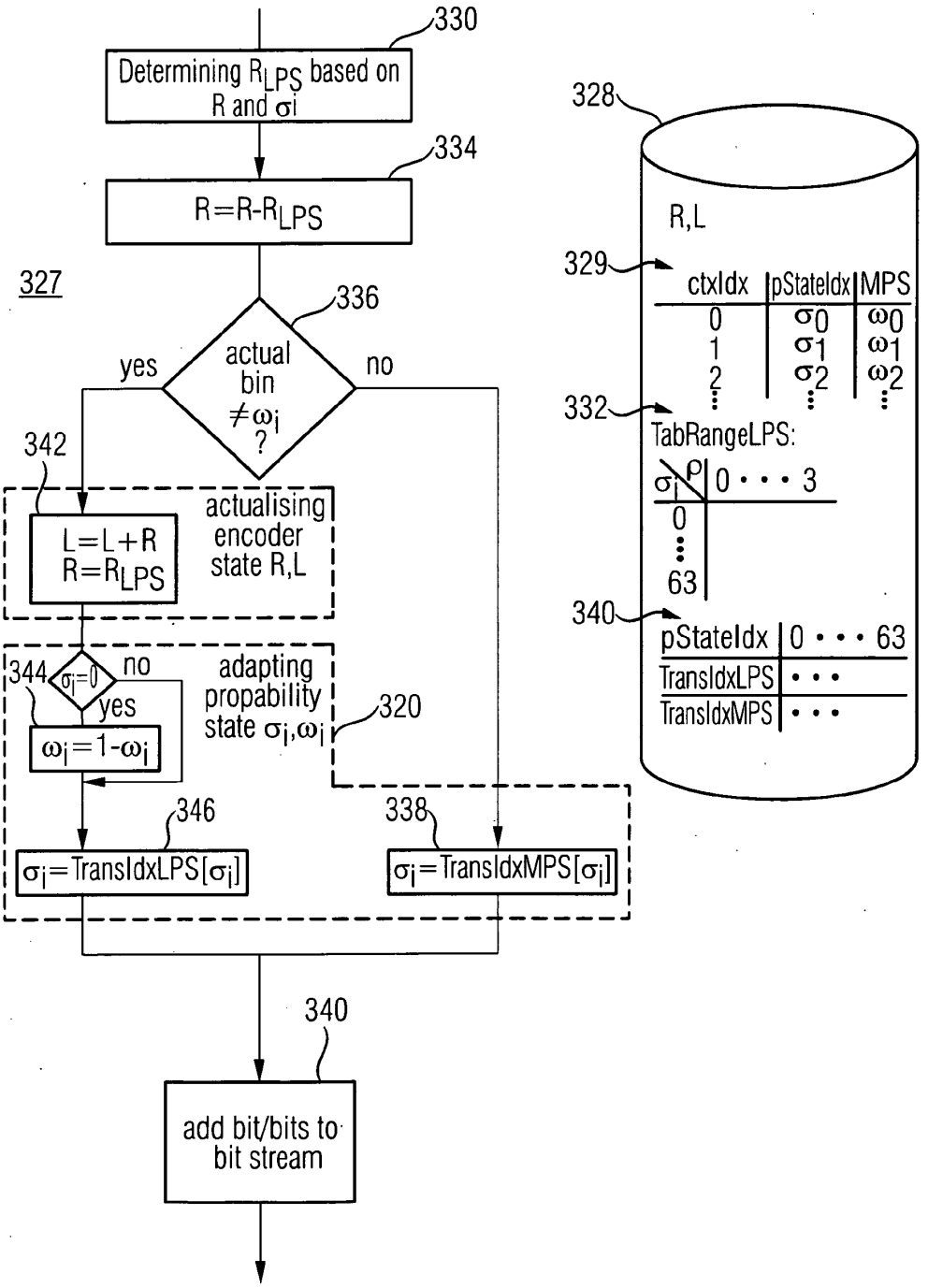

330
Determining R_{LPS} based on R and σ_i
334
$R=R-R_{LPS}$
327
336
actual bin $\neq\omega_i$?
yes
no
342
$L=L+R$
$R=R_{LPS}$
actualising encoder state R,L
344
$\sigma_i=0$
no
yes
$\omega_i=1-\omega_i$
adapting propability state σ_i,ω_i
320
346
$\sigma_i=TransIdxLPS[\sigma_i]$
338
$\sigma_i=TransIdxMPS[\sigma_i]$
340
add bit/bits to bit stream
328
R,L
329
ctxIdx | pStateIdx | MPS
0 | σ_0 | ω_0
1 | σ_1 | ω_1
2 | σ_2 | ω_2
332
TabRangeLPS:
σ_i \ ρ | 0 • • • 3
0
63
340
pStateIdx | 0 • • • 63
TransIdxLPS | • • •
TransIdxMPS | • • •

FIG 6

xN	yN	currMbFrameFlag	mbIsTopMbFlag	mbAddrX	mbAddrXFrameFlag	additional condition	mbAddrN	yM
<0	0...16	1	1	mbAddrA	1		mbAddrA	yN
					0	yN % 2==0	mbAddrA	yN>>1
						yN % 2 !=0	mbAddrA+1	yN>>1
			0	mbAddrA	1		mbAddrA+1	yN
					0	yN % 2==0	mbAddrA	(yN+8)>>1
						yN % 2 !=0	mbAddrA+1	(yN+8)>>1
		0	1	mbAddrA	1	yN<8	mbAddrA	yN<<1
						yN>=8	mbAddrA+1	(yN<<1)-8
					0		mbAddrA	yN
			0	mbAddrA	1	yN<8	mbAddrA	(yN<<1)+1
						yN>=8	mbAddrA+1	(yN<<1)+1-8
					0		mbAddrA+1	yN
0...16	<0	1	1	mbAddrB			mbAddrB+1	yN
			0	CurrMbAddr			CurrMbAddr-1	yN
		0	1	mbAddrB	1		mbAddrB+1	2*yN
					0		mbAddrB	yN
			0	mbAddrB			mbAddrB+1	yN
0...16	0...16			CurrMbAddr			CurrMbAddr	yN

FIG 8

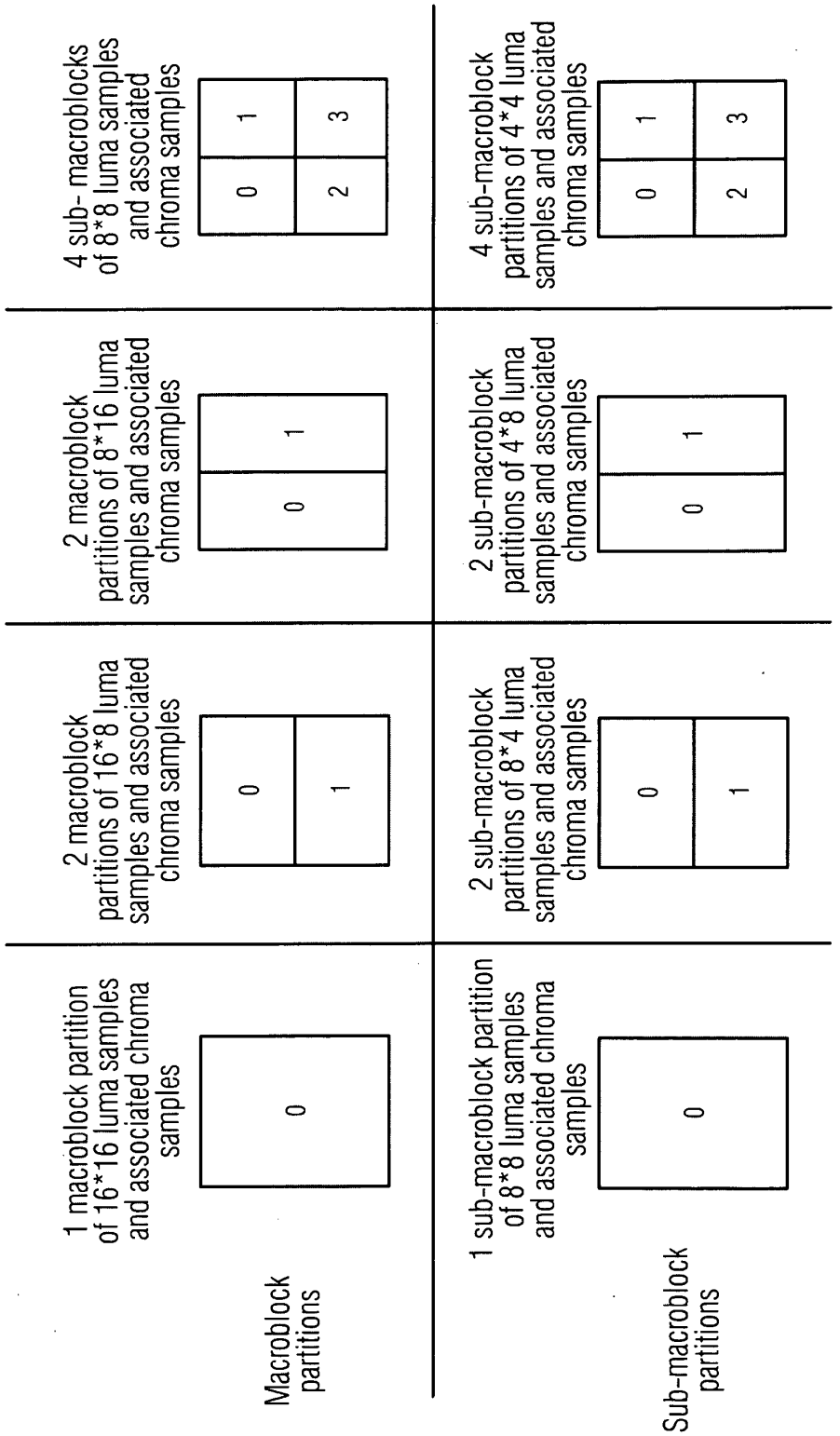

Macroblock partitions
1 macroblock partition of 16*16 luma samples and associated chroma samples
0
2 macroblock partitions of 16*8 luma samples and associated chroma samples
0
1
2 macroblock partitions of 8*16 luma samples and associated chroma samples
0
1
4 sub- macroblocks of 8*8 luma samples and associated chroma samples
0
1
2
3
Sub-macroblock partitions
1 sub-macroblock partition of 8*8 luma samples and associated chroma samples
0
2 sub-macroblock partitions of 8*4 luma samples and associated chroma samples
0
1
2 sub-macroblock partitions of 4*8 luma samples and associated chroma samples
0
1
4 sub-macroblock partitions of 4*4 luma samples and associated chroma samples
0
1
2
3

FIG 9

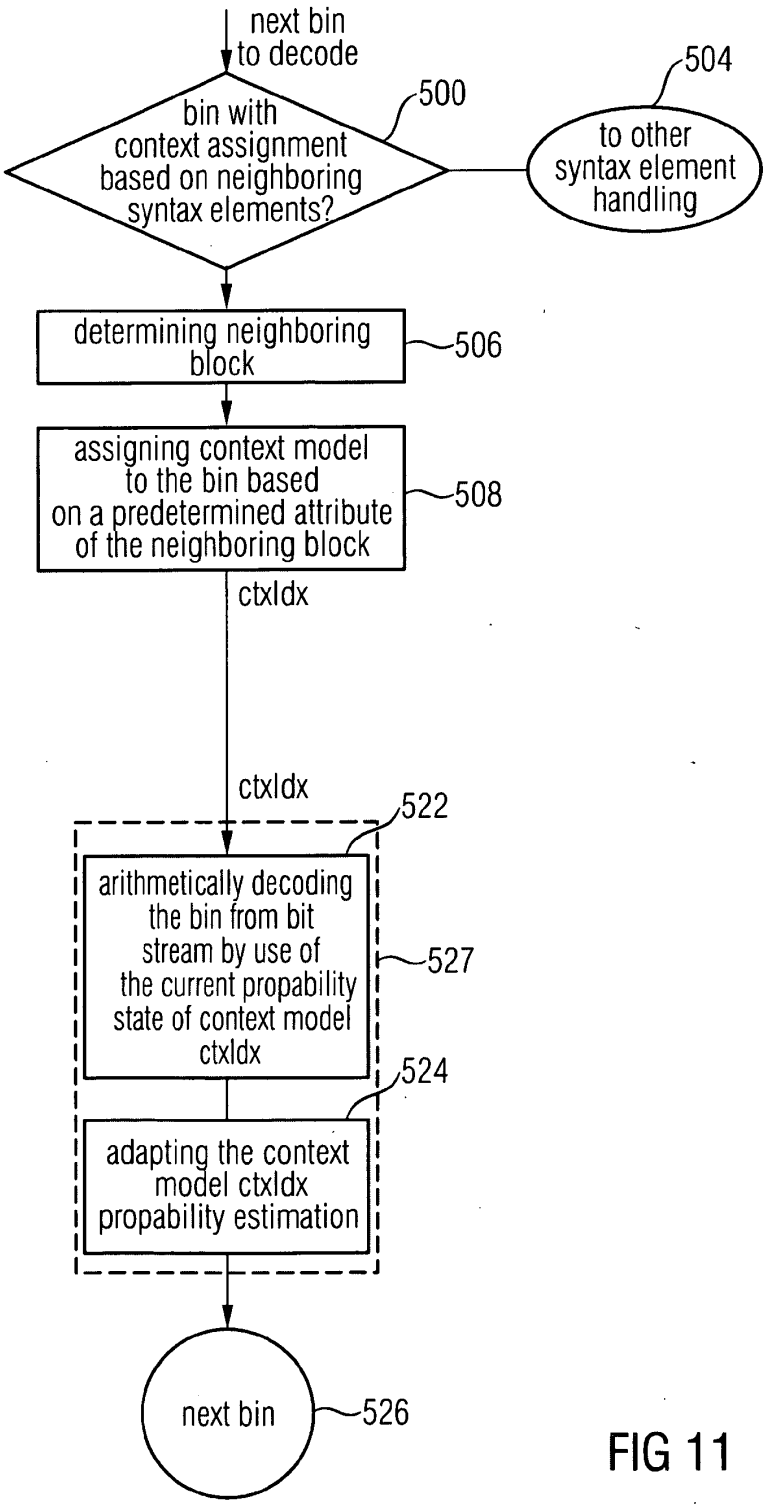

next bin
to decode
bin with
context assignment
based on neighboring
syntax elements?
500
504
to other
syntax element
handling
determining neighboring
block
506
assigning context model
to the bin based
on a predetermined attribute
of the neighboring block
508
ctxIdx
ctxIdx
522
arithmetically decoding
the bin from bit
stream by use of
the current propability
state of context model
ctxIdx
527
524
adapting the context
model ctxIdx
propability estimation
next bin
526

FIG 11

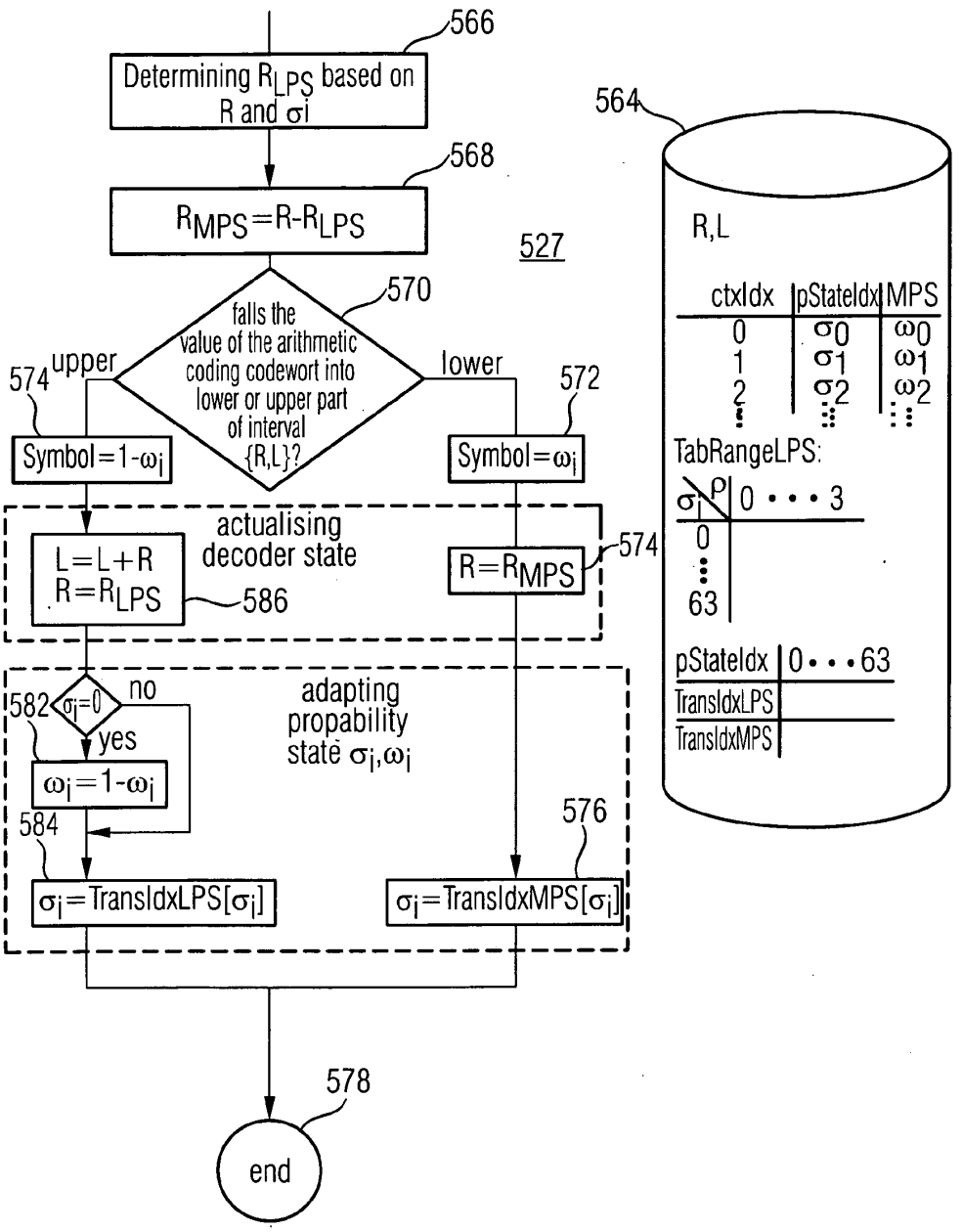

566
Determining R_{LPS} based on R and σi
568
$R_{MPS}=R-R_{LPS}$
527
570
falls the value of the arithmetic coding codewort into lower or upper part of interval {R,L}?
upper
lower
574
572
Symbol=1-ωi
Symbol=ωi
actualising decoder state
L=L+R
$R=R_{LPS}$
586
$R=R_{MPS}$
574
582
σi=0
no
yes
adapting propability state σi,ωi
ωi=1-ωi
584
576
σi=TransIdxLPS[σi]
σi=TransIdxMPS[σi]
578
end
564
R,L
ctxIdx
pStateIdx
MPS
0
1
2
σ0
σ1
σ2
ω0
ω1
ω2
TabRangeLPS:
ρ
0 • • • 3
0
63
pStateIdx
0 • • • 63
TransIdxLPS
TransIdxMPS

FIG 12

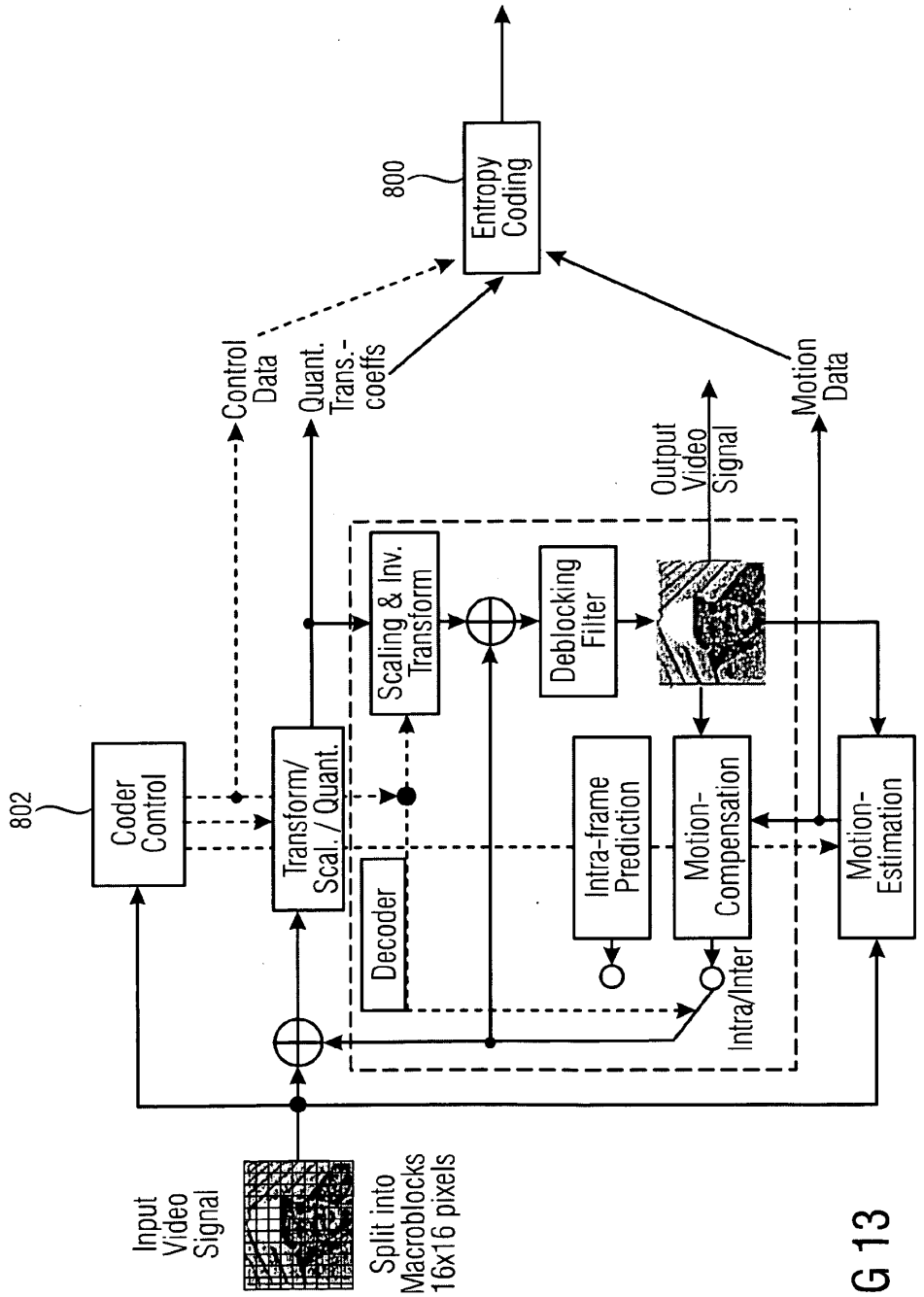

Input Video Signal
Split into Macroblocks 16x16 pixels
802
Coder Control
Transform/ Scal. / Quant.
Decoder
Scaling & Inv. Transform
Deblocking Filter
Intra-frame Prediction
Motion-Compensation
Intra/Inter
Motion-Estimation
Control Data
Quant. Trans.-coeffs
800
Entropy Coding
Output Video Signal
Motion Data

FIG 13

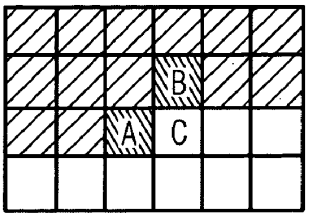


FIG 14

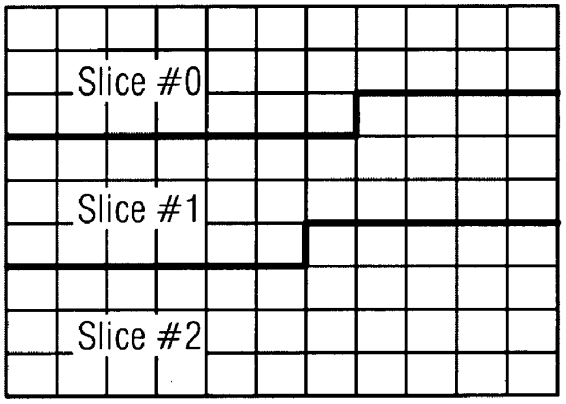


FIG 15

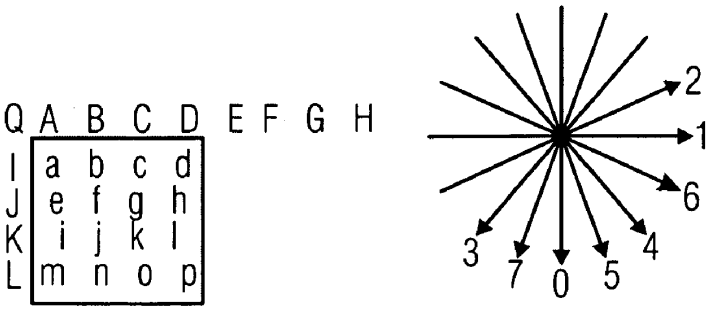


FIG 16

VIDEO FRAME ENCODING AND DECODING

[0001] This application is a divisional of utility application Ser. No. 10/769,403, filed Jan. 30, 2004, the entirety of which is herein incorporated by this reference thereto.

BACKGROUND OF THE INVENTION

[0002] I. Technical Field of the Invention

[0003] The present invention is related to video frame coding and, in particular, to an arithmetic coding scheme using context assignment based on neighboring syntax elements.

[0004] II. Description of the Prior Art

[0005] Entropy coders map an input bit stream of binarizations of data values to an output bit stream, the output bit stream being compressed relative to the input bit stream, i.e., consisting of less bits than the input bit stream. This data compression is achieved by exploiting the redundancy in the information contained in the input bit stream.

[0006] Entropy coding is used in video coding applications. Natural camera-view video signals show non-stationary statistical behavior. The statistics of these signals largely depend on the video content and the acquisition process. Traditional concepts of video coding that rely on mapping from the video signal to a bit stream of variable length-coded syntax elements exploit some of the non-stationary characteristics but certainly not all of it. Moreover, higher-order statistical dependencies on a syntax element level are mostly neglected in existing video coding schemes. Designing an entropy coding scheme for video coder by taking into consideration these typical observed statistical properties, however, offer significant improvements in coding efficiency.

[0007] Entropy coding in today's hybrid block-based video coding standards such as MPEG-2 and MPEG-4 is generally based on fixed tables of variable length codes (VLC). For coding the residual data in these video coding standards, a block of transform coefficient levels is first mapped into a one-dimensional list using an inverse scanning pattern. This list of transform coefficient levels is then coded using a combination of run-length and variable length coding. The set of fixed VLC tables does not allow an adaptation to the actual symbol statistics, which may vary over space and time as well as for different source material and coding conditions. Finally, since there is a fixed assignment of VLC tables and syntax elements, existing inter-symbol redundancies cannot be exploited within these coding schemes.

[0008] It is known, that this deficiency of Huffman codes can be resolved by arithmetic codes. In arithmetic codes, each symbol is associated with a respective probability value, the probability values for all symbols defining a probability estimation. A code word is coded in an arithmetic code bit stream by dividing an actual probability interval on the basis of the probability estimation in several sub-intervals, each sub-interval being associated with a possible symbol, and reducing the actual probability interval to the sub-interval associated with the symbol of data value to be coded. The arithmetic code defines the resulting interval limits or some probability value inside the resulting probability interval.

[0009] As may be clear from the above, the compression effectiveness of an arithmetic coder strongly depends on the probability estimation as well as the symbols, which the probability estimation is defined on.

[0010] A special kind of context-based adaptive binary arithmetic coding, called CABAC, is employed in the H.264/AVC video coding standard. There was an option to use macroblock adaptive frame/field (MBAFF) coding for interlaced video sources. Macroblocks are units into which the pixel samples of a video frame are grouped. The macroblocks, in turn, are grouped into macroblock pairs. Each macroblock pair assumes a certain area of the video frame or picture. Furthermore, several macroblocks are grouped into slices. Slices that are coded in MBAFF coding mode can contain both, macroblocks coded in frame mode and macroblocks coded in field mode. When coded in frame mode, a macroblock pair is spatially sub-divided into a top and a bottom macroblock, the top and the bottom macroblock comprising both pixel samples captured at a first time instant and picture samples captured at the second time instant being different from the first time instant. When coded in field mode, the pixel samples of a macroblock pair are distributed to the top and the bottom macroblock of the macroblock pair in accordance with their capture time.

[0011] The introduction of MBAFF coding to the precoding stage as an alternative to PAFF (picture adaptive frame/field) coding where the decisions between frame and field coding are made for each frame as a hole, was motivated by the fact that if a frame consists of mixed regions where some regions are moving and others are not, it is typically more efficient to code the non-moving regions in frame mode and the moving regions in the field mode.

[0012] As mentioned above, in the H.264/AVC video coding standard, there is an option to use macroblock adaptive frame/field coding (MBAFF) for interlaced video sources. As turned out from the above considerations, in MBAFF, the pixel samples in a respective macroblock pair are distributed in different ways to the top end field macroblock, depending on the macroblock pair being frame or field coded. Thus, on the one hand, when MBAFF mode is active, the neighborhood between pixel samples of neighboring is somewhat complicated compared to the case of PAFF coding mode.

[0013] On the other hand, the CABAC entropy coding scheme tries to exploit statistical redundancies between the values of syntax elements of neighboring blocks. That is, for the coding of the individual binary decisions, i.e., bins, of several syntax elements, context variables are assigned depending on the values of syntax elements of neighboring blocks located to the left of and above the current block. In this document, the term "block" is used as collective term that can represent 4×4 luma or chroma blocks used for transform coding, 8×8 luma blocks used for specifying the coded block pattern, macroblocks, macroblock or sub-macroblock partitions used for motion description.

[0014] In the case of macroblock adaptive frame/field coding, while the neighborhoods that are used for CABAC are not clear since field and frame macroblocks can be mixed inside the picture or slice. In the solution to this problem that was included in older versions of the H.264/AVC, each macroblock pair was considered as frame macroblock pair for the purpose of context modeling in CABAC. However, with this concept, the coding efficiency could be degraded, since choosing neighboring blocks that do not adjoin to the current blocks affects the adaption of the conditional probability models.

SUMMARY OF THE INVENTION

[0015] It is the object of the present invention to provide a video coding scheme, which enables a higher compression effectiveness.

[0016] In accordance with the first aspect of the present invention, this object is achieved by a method for encoding a video signal representing at least one video frame, with at least one video frame being composed of picture samples, the picture samples belonging either to a first or a second field being captured at different time instants, the video frame being spatially divided up into macroblock pair regions, each macroblock pair region being associated with a top and bottom macroblock, the method comprising the steps of deciding, for each macroblock pair region, as to whether same is of a first or a second distribution type; assigning, for each macroblock pair region, each of the pixel samples in the respective macroblock pair region to a respective one of the top and bottom macroblock of the respective macroblock pair region, in accordance with the distribution type of the respective macroblock pair region, and pre-coding the video signal into a pre-coded video signal, the pre-coding comprising the sub-step of pre-coding a current macroblock of the top and bottom macroblock associated with a current macroblock pair region of the macroblock pair regions to obtain a current syntax element. Thereafter, it is determined, for the current syntax element, a neighboring macroblock at least based upon as to whether the current macroblock pair region is of a first or second distribution type. One of at least two context models is assigned to the current syntax element based on a pre-determined attribute of the neighboring macroblock, wherein each context model is associated with a different probability estimation. Finally, arithmetically encoding the syntax element into a coded bit stream based on the probability estimation with which the assigned context model is associated.

[0017] In accordance with the second aspect of the present invention, this object is achieved by a method for decoding a syntax element from a coded bit stream, the coded bit stream being an arithmetically encoded version of a pre-coded video signal, the pre-coded video signal being a pre-coded version of a video signal, the video signal representing at least one video frame being composed of picture samples, the picture samples belonging either to a first or a second field being captured at a different time instants, the video frame being spatially divided up into macroblock pair regions, each macroblock pair region being associated with a top and a bottom macroblock, each macroblock pair region being either of a first or a second distribution type, wherein, for each macroblock pair region, each of the pixel samples in the respective macroblock pair region is assigned to a respective one of the top and bottom macroblock of the respective macroblock pair region in accordance with the distribution type of the respective macroblock pair region, wherein the syntax element relates to a current macroblock of the top and bottom macroblock of a current macroblock pair region of the macroblock pair regions. The method comprises determining, for the current syntax element, a neighboring macroblock at least based upon as to whether the current macroblock pair region is of a first or a second distribution type; assigning one of at least two context models to the current syntax element based on a predetermined attribute of the neighboring macroblock, wherein each context model is associated with a different probability estimation; and arithmetically decoding the syntax element from the coded bit stream based on the probability estimation with which the assigned context model is associated.

[0018] In accordance with the third aspect of the present invention, this object is achieved by an Apparatus for encoding a video signal representing at least one video frame, with at least one video frame being composed of picture samples, the picture samples belonging either to a first or a second field being captured at different time instants, the video frame being spatially divided up into macroblock pair regions, each macroblock pair region being associated with a top and bottom macroblock, the apparatus comprising means for deciding, for each macroblock pair region, as to whether same is of a first or a second distribution type; means for assigning, for each macroblock pair region, each of the pixel samples in the respective macroblock pair region to a respective one of the top and bottom macroblock of the respective macroblock pair region, in accordance with the distribution type of the respective macroblock pair region; means for pre-coding the video signal into a pre-coded video signal, the pre-coding comprising the sub-step of pre-coding a current macroblock of the top and bottom macroblock associated with a current macroblock pair region of the macroblock pair regions to obtain a current syntax element; means for determining, for the current syntax element, a neighboring macroblock at least based upon as to whether the current macroblock pair region is of a first or second distribution type; means for assigning one of at least two context models to the current syntax element based on a pre-determined attribute of the neighboring macroblock, wherein each context model is associated with a different probability estimation; and means for arithmetically encoding the syntax element into a coded bit stream based on the probability estimation with which the assigned context model is associated.

[0019] In accordance with the forth aspect of the present invention, this object is achieved by an apparatus method for decoding a syntax element from a coded bit stream, the coded bit stream being an arithmetically encoded version of a pre-coded video signal, the pre-coded video signal being a pre-coded version of a video signal, the video signal representing at least one video frame being composed of picture samples, the picture samples belonging either to a first or a second field being captured at a different time instants, the video frame being spatially divided up into macroblock pair regions, each macroblock pair region being associated with a top and a bottom macroblock, each macroblock pair region being either of a first or a second distribution type, wherein, for each macroblock pair region, each of the pixel samples in the respective macroblock pair region is assigned to a respective one of the top and bottom macroblock of the respective macroblock pair region in accordance with the distribution type of the respective macroblock pair region, wherein the syntax element relates to a current macroblock of the top and bottom macroblock of a current macroblock pair region of the macroblock pair regions, wherein the apparatus comprises means for determining, for the current syntax element, a neighboring macroblock at least based upon as to whether the current macroblock pair region is of a first or a second distribution type; means for assigning one of at least two context models to the current syntax element based on a predetermined attribute of the neighboring macroblock, wherein each context model is associated with a different probability estimation; and

[0020] mean for arithmetically decoding the syntax element from the coded bit stream based on the probability estimation with which the assigned context model is associated.

[0021] The present invention is based on the finding that when, for whatever reason, such as the better effectiveness when coding video frames having non-moving regions and

moving regions, macroblock pair regions of a first and a second distribution type, i.e., field and frame coded macroblock pairs, are used concurrently in a video frame, i.e. MBAFF coding is used, the neighborhood between contiguous blocks of pixel samples has to be defined in a way different from considering each macroblock pair as frame macroblock pair for the purpose of context modeling and that the distance of areas covered by a neighboring and a current block could be very large when considering each macroblock pair as a frame macroblock pair. This in turn, could degrade the coding efficiency, since choosing neighboring blocks that are not arranged nearby the current block affects the adaption of the conditional probability models.

[0022] Further, the present invention is based on the finding, that, for sake of a better definition of neighborhood between blocks of picture samples, i.e. the neighboring block which the syntax element to be coded or decoded relates to and the current block based on the attribute of which the assignment of a context model is conducted, and when the neighboring block lies beyond the borders or circumference of the current macroblock containing the current block, it is important to make the determination of the macroblock containing the neighboring block dependent upon as to whether the current macroblock pair region containing the current block is of a first or a second distribution type, i.e., frame or field coded.

[0023] The blocks may be a macroblock or some sub-part thereof. In both cases, the determination of a neighboring block comprises at least the determination of a neighboring macroblock as long as the neighboring block lies beyond the borders of the current macroblock.

SHORT DESCRIPTION OF THE DRAWINGS

[0024] Preferred embodiments of the present invention are described in more detail below with respect to the figures.

[0025] FIG. **1** shows a high-level block diagram of a coding environment in which the present invention may be employed.

[0026] FIG. **2** shows a block diagram of the entropy coding part of the coding environment of FIG. **1**, in accordance with an embodiment of the present invention.

[0027] FIG. **3** shows a schematic diagram illustrating the spatial subdivision of a picture or video frame into macroblock pairs, in accordance with an embodiment of the present invention.

[0028] FIG. **4***a* shows a schematic diagram illustrating the frame mode, in accordance with an embodiment of the present invention.

[0029] FIG. **4***b* shows a schematic diagram illustrating the field mode, in accordance with an embodiment of the present invention.

[0030] FIG. **5** shows a flow diagram illustrating the encoding of syntax elements with context assignments based on neighboring syntax elements in accordance with an embodiment of the present invention.

[0031] FIG. **6** shows a flow diagram illustrating the binary arithmetic coding of the syntax elements based on the context model to which it is assigned in accordance with an embodiment of the present invention.

[0032] FIG. **7** shows a schematic diagram illustrating the addressing scheme of the macroblocks in accordance with an embodiment of the present invention.

[0033] FIG. **8** shows a table illustrating how to obtain the macroblock address mbAddrN indicating the macroblock containing a sample having coordinates xN and yN relative to the upper-left sample of a current macroblock and, additionally, the y coordinate yM for the sample in the macroblock mbAddrN for that sample, dependent on the sample being arranged beyond the top or the left border of the current macroblock, the current macroblock being frame or field coded, and the current macroblock being the top or the bottom macroblock of the current macroblock pair, and, eventually, the macroblock mbAddrA being frame or field coded and the line in which the sample lies having an odd or even line number yN.

[0034] FIG. **9** shows a schematic illustrating macroblock partitions, sub-macroblock partitions, macroblock partitions scans, and sub-macroblock partition scans.

[0035] FIG. **10** shows a high-level block diagram of a decoding environment in which the present invention may be employed.

[0036] FIG. **11** shows a flow diagram illustrating the decoding of the syntax elements coded as shown in FIGS. **5** and **6** from the coded bit stream, in accordance with an embodiment of the present invention.

[0037] FIG. **12** shows a flow diagram illustrating the arithmetical decoding process and the decoding process of FIG. **11** in accordance with an embodiment of the present invention.

[0038] FIG. **13** shows a basic coding structure for the emerging H.264/AVC video encoder for a macroblock.

[0039] FIG. **14** illustrates a context template consisting of two neighboring syntax elements A and B to the left and on the top of the current syntax element C.

[0040] FIG. **15** shows an illustration of the subdivision of a picture into slices.

[0041] FIG. **16** shows, to the left, intra_4×4 prediction conducted for samples a-p of a block using samples A_Q, and to the right, "prediction directions for intra_4×4 prediction.

[0042] FIG. **1** shows a general view of a video encoder environment to which the present invention could be applied. A picture of video frame **10** is fed to a video precoder **12**. The video precoder treats the picture **10** in units of so-called macroblocks **10***a*. Each macroblock contains several picture samples of picture **10**. On each macroblock a transformation into transformation coefficients is performed followed by a quantization into transform coefficient levels. Moreover, intra-frame prediction or motion compensation is used in order not to perform the afore mentioned steps directly on the pixel data but on the differences of same to predicted pixel values, thereby achieving small values which are more easily compressed.

[0043] Precoder **12** outputs the result, i.e., the precoded video signal. All residual data elements in the precoded video signal, which are related to the coding of transform coefficients, such as the transform coefficient levels or a significance map indicating transform coefficient levels skipped, are called residual data syntax elements. Besides these residual data syntax elements, the precoded video signal output by precoder **12** contains control information syntax elements containing control information as to how each macroblock has been coded and has to be decoded, respectively. In other words, the syntax elements are dividable into two categories. The first category, the control information syntax elements, contains the elements related to a macroblock type, sub-macroblock type, and information on prediction modes both of a spatial and of temporal types as well as slice-based and macroblock-based control information, for example. In the second category, all residual data elements such as a signifi-

cance map indicating the locations of all significant coefficients inside a block of quantized transform coefficients, and the values of the significant coefficients, which are indicated in units of levels corresponding to the quantizations steps, are combined, i.e., the residual data syntax elements.

[0044] The macroblocks into which the picture **10** is partitioned are grouped into several slices. In other words, the picture **10** is subdivided into slices. An example for such a subdivision is shown in FIG. **16**, in which each block or rectangle represents a macroblock. For each slice, a number of syntax elements are generated by precoder **12**, which form a coded version of the macro blocks of the respective slice.

[0045] The precoder **12** transfers the syntax elements to a final coder stage **14**, which is an entropy coder and explained in more detail with respect to FIG. **2**. The final coder stage **14** generates an arithmetic codeword for each slice. When generating the arithmetic codeword for a slice, the final coding stage **14** exploits the fact that each syntax element is a data value having a certain meaning in the video signal bit stream that is passed to the entropy coder **14**. The entropy coder **14** outputs a final compressed arithmetic code video bit stream comprising arithmetic codewords for the slices of picture **10**.

[0046] FIG. **2** shows the arrangement for coding the syntax elements into the final arithmetic code bit stream, the arrangement generally indicated by reference number **100**. The coding arrangement **100** is divided into three stages, **10***a*, **10***b*, and **100***c*.

[0047] The first stage **100***a* is the binarization stage and comprises a binarizer **102**. An input of the binarizer **102** is connected to an input **104** of stage **100***a* via a switch **106**. At the same time, input **104** forms the input of coding arrangement **100**. The output of binarizer **102** is connected to an output **108** of stage **100***a*, which, at the same time, forms the input of stage **10***b*. Switch **106** is able to pass syntax elements arriving at input **104** to either binarizer **102** or binarization stage output **108**, thereby bypassing binarizer **102**.

[0048] The function of switch **106** is to directly pass the actual syntax element at input **104** to the binarization stage output **108** if the syntax element is already in a wanted binarized form. Examples for syntax elements that are not in the correct binarization form, called non-binary valued syntax elements, are motion vector differences and transform coefficient levels. Examples for a syntax element that has not to be binarized since it is already a binary value comprise the MBAFF (MBAFF=Macroblock Adaptive Frame/Field) Coding mode flag or mb_field_decoding_flag, the mb_skip_flag and coded_block_flag to be described later in more detail. Examples for a syntax element that has to be binarized since it is not a binary value comprise syntax elements mb_type, coded_block_pattern, ref_idx_l0, ref_idx_l1, mvd_l0, mvd_l1, and intro_chroma_pred_mode.

[0049] Different binarization schemes are used for the syntax elements to be binarized. For example, a fixed-length binarization process is constructed by using an L-bit unsigned integer bin string of the syntax element value, where L is equal to $\log_2(cMax+1)$ rounded up to the nearest integer greater than or equal to the sum, with cMax being the maximum possible value of the syntax element. The indexing of the bins for the fl binarization is such that the bin index of zero relates to the least significant bit with increasing values of the bin index towards the most significant bit. Another binarization scheme is a truncated unary binarization scheme where syntax element values C smaller than the largest possible value cMax are mapped to a bit or bin string of length C+1 with the bins having a bin index smaller than C being equal to 1 and the bin having the bin index of C being equal to 0, whereas for syntax elements equal to the largest possible value cMax, the corresponding bin string is a bit string of length cMax with all bits equal to one not followed by a zero. Another binarization scheme is a k-th order exponential Golomb binarization scheme, where a syntax element is mapped to a bin string consisting of a prefix bit string and, eventually, a suffix bit string.

[0050] The non-binary valued syntax elements are passed via switch **106** to binarizer **102**. Binarizer **102** maps the non-binary valued syntax elements to a codeword, or a so-called bin string, so that they are now in a binary form. The term "bin" means the binary decision that have to be made at a node of a coding tree defining the binarization mapping of a non-binary value to a bit string or codeword, when transitioning from the route node of the coding tree to the leaf of the coding tree corresponding to the non-binary value of the non-binary syntax element to be binarized. Thus, a bin string is a sequence of bins or binary decisions and corresponds to a codeword having the same number of bits, each bit being the result of a binary decision.

[0051] The bin strings output by binarizer **102** may not be passed directly to binarization stage output **108** but controllably passed to output **108** by a bin loop over means **110** arranged between the output of binarizer **102** and output **108** in order to merge the bin strings output by binarizer **102** and the already binary valued syntax elements bypassing binarizer **102** to a single bit stream at binarization stage output **108**.

[0052] Thus, the binarization stage **108** is for transferring the syntax elements into a suitable binarized representation. The binarization procedure in binarizer **102** preferably yields a binarized representation which is adapted to the probability distribution of the syntax elements so as to enable very efficient binary arithmetic coding.

[0053] Stage **100***b* is a context modeling stage and comprises a context modeler **112** as well as a switch **113**. The context modeler **112** comprises an input, an output, and an optional feedback input. The input of context modeler **112** is connected to the binarization stage output **108** via switch **113**. The output of context modeler **112** is connected to a regular coding input terminal **114** of stage **100***c*. The function of switch **113** is to pass the bits or bins of the bin sequence at binarization stage output **108** to either the context modeler **112** or to a bypass coding input terminal **116** of stage **100***c*, thereby bypassing context modeler **112**.

[0054] The aim of switch **113** is to ease the subsequent binary arithmetic coding performed in stage **100***c*. To be more precise, some of the bins in the bin string output by binarizer **102** show heuristically nearly an equi-probable distribution. This means, the corresponding bits are, with a probability of nearly 50%, 1 and, with a probability of nearly 50%, 0, or, in other words, the bits corresponding to this bin in a bin string have a 50/50 chance to be 1 or 0. These bins are fed to the bypass-coding input terminal **116** and are binary arithmetically coded by use of an equi-probable probability estimation, which is constant and, therefore, needs no adaption or updating overhead. For all other bins, it has been heuristically determined that the probability distribution of these bins depends on other bins as output by stage **10***a* so that it is worthwhile to adapt or update the probability estimation used for binary arithmetically coding of the respective bin as it will be described in more detail below exemplarily with respect to

exemplary syntax elements. The latter bins are thus fed by switch **113** to the input terminal of context modeler **112**.

[0055] Context modeler **112** manages a set of context models. For each context model, the context modeler **112** has stored an actual bit or bin value probability distribution estimation. For each bin that arrives at the input of context modeler **112**, the context modeler **112** selects one of the sets of context models. In other words, the context modeler **112** assigns the bin to one of the set of context models. The assignment of bins to a context model is such that the actual probability distribution of bins belonging to the same context model show the same or likewise behavior so that the actual bit or bin value probability distribution estimation stored in the context modeler **112** for a certain context model is a good approximation of the actual probability distribution for all bins that are assigned to this context model. The assignment process in accordance with the present invention exploits the spatial relationship between syntax element of neighboring blocks. This assignment process will be described in more detail below.

[0056] When having assigned the context model to an incoming bin the context modeler **112** passes the bin further to arithmetical coding stage **100***c* together with the probability distribution estimation of the context model, which the bin is assigned to. By this measure, the context modeler **112** drives the arithmetical coding stage **100***c* to generate a sequence of bits as a coded representation of the bins input in context modeler **112** by switch **113** according to the switched bit value probability distribution estimations as indicated by the context modeler **112**.

[0057] Moreover, the context modeler **112** continuously updates the probability distribution estimations for each context model in order to adapt the probability distribution estimation for each context model to the property or attributes of the picture or video frame from which the syntax elements and bins have been derived. The estimation adaptation or estimation update is based on past or prior bits or bin values which the context modeler **112** receives at the feedback input over a feedback line **117** from stage **100***c* or may temporarily store. Thus, in other words, the context modeler **112** updates the probability estimations in response to the bin values passed to arithmetical coding stage **100***c*. To be more precise, the context modeler **112** uses a bin value assigned to a certain context model merely for adaptation or update of the probability estimation that is associated with the context model of this bin value.

[0058] Some of the syntax elements, when the same bin or same syntax element occurs several times in the bins passed from stage **100***a* may be assigned to different of the context models each time they occur, depending on previously incoming or previously arithmetically coded bins, and/or depending on other circumstances, such as previously coded syntax elements of neighboring blocks, as is described in more detail below with respect to exemplary syntax elements.

[0059] It is clear from the above, that the probability estimation used for binary arithmetically coding determines the code and its efficiency in the first place, and that it is of paramount importance to have an adequate model that exploits the statistical dependencies of the syntax elements and bins to a large degree so that the probability estimation is always approximating very effectively the actual probability distribution during encoding.

[0060] The third stage **100***c* of coding arrangement **100** is the arithmetic coding stage. It comprises a regular coding engine **118**, a bypass-coding engine **120**, and a switch **122**. The regular coding engine **118** comprises an input and an output terminal. The input terminal of regular coding engine **118** is connected to the regular coding input terminal **114**. The regular coding engine **118** binary arithmetically codes the bin values passed from context modeler **112** by use of the context model also passed from context modeler **112** and outputs coded bits. Further, the regular coding engine **118** passes bin values for context model updates to the feedback input of context modeler **112** over feedback line **117**.

[0061] The bypass-coding engine **112** has also an input and an output terminal, the input terminal being connected to the bypass coding input terminal **116**. The bypass-coding engine **120** is for binary arithmetically coding the bin values passed directly from binarization stage output **108** via switch **113** by use of a static predetermined probability distribution estimation and also outputs coded bits.

[0062] The coded bits output from regular coding engine **118** and bypass coding engine **120** are merged to a single bit stream at an output **124** of coding arrangement **100** by switch **122**, the bit stream representing a binary arithmetic coded bit stream of the syntax elements as input in input terminal **104**. Thus, regular coding engine **118** and bypass coding **120** cooperate in order to bit wise perform arithmetical coding based on either an adaptive or a static probability distribution model.

[0063] After having described with respect to FIGS. **1** and **2** rather generally the operation of coding arrangement **100**, in the following its functioning is described in more detail with respect to the handling of exemplary syntax elements for which an context assignment process based on syntax elements of neighboring blocks is used, in accordance with embodiments of the present invention. In order to do so, firstly, with regard to FIGS. **3** to **4***b*, the meaning of MBAFF coding is described, in order to enable a better understanding of the definition of neighborhood between a current block and a neighboring block used during assignment of a context model to a syntax element concerning the current block in case of MBAFF.

[0064] FIG. **3** shows a picture or decoded video frame **10**. The video frame **10** is spatially partitioned into macroblock pairs **10***b*. The macroblock pairs are arranged in an array of rows **200** and columns **202**. Each macroblock pair consists of two macroblocks **10***a*.

[0065] In order to be able to address each macroblock **10***a*, a sequence is defined with respect to macroblocks **10***a*. In order to do so, in each macroblock pair, one macroblock is designated the top macroblock whereas the other macroblock in the macroblock pair is designated the bottom macroblock, the meaning of top and bottom macroblock depending on the mode by which a macroblock pair is coded by precoder **12** (FIG. **1**) as will be described with respect to FIGS. **4***a* and **4***b*. Thus, each macroblock pair row **200** consists of two macroblock rows, i.e., an top macroblock row **200***a* consisting of the top macroblocks in the macroblock pairs of the macroblock pair line **200** and a bottom macroblock row **200***b* comprising the bottom macroblocks of the macroblock pairs.

[0066] In accordance with the present example, the top macroblock of the top left macroblock pair resides at address zero. The next address, i.e. address **1**, is assigned to the bottom macroblock of the top left macroblock pair. The addresses of the top macroblocks of the macroblock pairs in the same, i.e., top macroblock row **200***a*, are 2, 4, . . . , 2i−2, with the addresses rising from left to right, and with i expressing the picture width in units of macroblocks or macroblock

pairs. The addresses 1, 3, . . . , 2i−1 are assigned to the bottom macroblocks of the macroblock pairs in the top macroblock pair row **200**, the addresses rising from left to right. The next 2i-addresses from 2i to 4i−1 are assigned to the macroblocks of the macroblock pairs in the next macroblock pair row from the top and so on, as illustrated in FIG. **3** by the numbers written into the boxes representing the macroblocks **10***a* and by the arched rows.

[0067] It is emphasized that FIG. **3** does show the spatial subdivision of picture **10** in units of macroblock pairs rather than in macroblocks. Each macroblock pair **10***b* represents a spatial rectangular region of the pictures. All picture samples or pixels (not shown) of picture **10** lying in the spatial rectangular region of a specific macroblock pair **10***b* belong to this macroblock pair. If a specific pixel or picture sample belongs to the top or the bottom macroblock of a macroblock pair depends on the mode by which precoder **12** has coded the macroblocks in that macroblock pair as it is described in more detail below.

[0068] FIG. **4***a* shows on the left hand side the arrangement of pixels or picture samples belonging to a macroblock pair **10***b*. As can be seen, the pixels are arranged in an array of rows and columns. Each pixel shown is indicated by a number in order to ease the following description of FIG. **4***a*. As can be seen in FIG. **4***a*, some of the pixels are marked by an "x" while the others are marked "□". All pixels marked with "x" belong to a first field of the picture while the other pixels marked with "□" belong to a second field of the picture. Pixels belonging to the same field are arranged in alternate rows of the picture. The picture or video frame can be considered to contain two interleaved fields, a top and a bottom field. The top field comprises the pixels marked with "□" and contains even-numbered rows 2n+2, 2n+4, 2n+6, . . . with 2n being the number of rows of one picture or video frame and n being an integer greater than or equal to 0. The bottom field contains the odd-numbered rows starting with the second line of the frame.

[0069] It is assumed that the video frame to which macroblock pair **10***b* belongs, is an interlaced frame where the two fields were captured at different time instants, for example the top field before the bottom field. It is now that the pixels or picture samples of a macroblock pair are differently assigned to the top or bottom macroblock of the macroblock pair, depending on the mode by which the respective macroblock pair is precoded by precoder **12** (FIG. **1**). The reason for this being the following.

[0070] As described above with respect to FIG. **1**, the picture samples of a macroblock, which may be luminance or luma and chrominance or chroma samples, may be either spatially or temporarily predicted by precoder **12**, and the resulting prediction residual is encoded using transform coding in order to yield the residual data syntax elements. It is now that in interlaced frames (and it is assumed that the present video frame is an interlaced frame), with regions of moving objects or camera motion, two adjacent rows of pixels tend to show a reduced degree of statistical dependency when compared to progressive video frames in which both fields are captured at the same time instant. Thus, in cases of such moving objects or camera motion, the pre-coding performed by precoder **12** which, as stated above, operates on macroblocks, may achieve merely a reduced compression efficiency when a macroblock pair is spatially sub-divided into a top macroblock representing the top half region of the macroblock pair and a bottom macroblock representing the bottom half region of the macroblock pair, since in this case, both macroblocks, the top and the bottom macroblock, comprise both top field and bottom field pixels. In this case, it may be more efficient for precoder **12** to code each field separately, i.e., to assign top field pixels to the top macroblock and bottom field pixels to the bottom field macroblock.

[0071] In order to illustrate as to how the pixels of a macroblock pair are assigned to the top and bottom macroblock of the, FIGS. **4***a* and **4***b* show on the right hand side the resulting top and bottom macroblock in accordance with the frame and field mode, respectively.

[0072] FIG. **4***a* represents the frame mode, i.e., where each macroblock pair is spatially subdivided in a top and a bottom half macroblock. FIG. **4***a* shows at **250** the top macroblock and at **252** the bottom macroblock as defined when they are coded in the frame mode, the frame mode being represented by double-headed arrow **254**. As can be seen, the top macroblock **250** comprises one half of the pixel samples of the macroblock pair **10***b* while the other picture samples are assigned to the bottom macroblock **252**. To be more specific, the picture samples of the top half rows numbered 2n+1 to 2n+6 belong to the top macroblock **250**, whereas the picture samples **91** to **96**, **101** to **106**, **111** to **116** of the bottom half comprising rows 2n+7 to 2n+12 of the macroblock pair **10***b* belong to the bottom macroblock **252**. Thus, when coded in frame mode, both macroblocks **250** and **252** comprise both, picture elements of the first field marked with "x" and captured at a first time instant and picture samples of the second field marked with "□" and captured at a second, different time instant.

[0073] The assignment of pixels as they are output by a camera or the like, to top or bottom macroblocks is slightly different in field mode. When coded in field mode, as is indicated by double headed arrow **256** in FIG. **4***b*, the top macroblock **252** of the macroblock pair **10***b* contains all picture samples of the top field, marked with "x", while the bottom macroblock **254** comprises all picture samples of the bottom field, marked with "□". Thus, when coded in accordance with field mode **256**, each macroblock in a macroblock pair does merely contain either picture samples of the top field or picture samples of the bottom field rather than a mix of picture samples of the top and bottom field.

[0074] Now, after having described the spatial sub-division of a picture into macroblock pairs and the assignment of picture samples in a macroblock pair to either the top or the bottom macroblock of the macroblock pair, the assignment depending on the mode by which the macroblock pair or the macroblocks of the macroblock pair are coded by precoder **12**, reference is again made to FIG. **1** in order to explain the function and meaning of the syntax element mb_field_decoding_flag contained in the precoded video signal output by precoder **12**, and, concurrently, in order to explain the advantages of MBAFF coded frames over just field or frame coded frames.

[0075] When the precoder **12** receives a video signal representing an interlaced video frame, precoder **12** is free to make the following decisions when coding the video frame **10**:

[0076] 1. It can combine the two fields together to code them as one single coded frame, so that each macroblock pair and each macroblock would be coded in frame mode.

[0077] 2. Alternatively, it could combine the two fields and code them as separate coded fields, so that each macroblock pair and each macroblock would be coded in field mode.

[0078] 3. As a last option, it could combine the two fields together and compress them as a single frame, but when coding the frame it splits the macroblock pairs into either pairs of two field macroblocks or pairs of two frame macroblocks before coding them.

[0079] The choice between the three options can be made adaptively for each frame in a sequence. The choice between the first two options is referred to as picture adaptive frame/field (PAFF) coding. When a frame is coded as two fields, each field is partitioned into macroblocks and is coded in a manner very similar to a frame.

[0080] If a frame consists of mixed regions where some regions are moving and others are not, it is typically more efficient to code the non-moving regions in frame mode and the moving regions in the field mode. Therefore, the frames/field encoding decision can be made independently for each vertical pair of macroblocks in a frame. This is the third coding option of the above-listed options. This coding option is referred to as macroblock adaptive frame/field (MBAFF) coding. It is assumed in the following that precoder **12** decides to use just this option. As described above, MBAFF coding allows the precoder to better adapt the coding mode type (filed or frame mode) to the respective areas of scenes. For example, precoder **12** codes macroblock pairs located at stationary areas of a video scene in frame mode, while coding macroblock pairs lying in areas of a scene showing fast movements in field mode.

[0081] As mentioned above, for a macroblock pair that is coded in frame mode, each macroblock contains frame lines. For a macroblock pair that is coded in field mode, the top macroblock contains top field lines and the bottom macroblock contains bottom field lines. The frame/field decision for each macroblock pair is made at the macroblock pair level by precoder **12**, i.e. if the top macroblock is field coded same applies for the bottom macroblock within same macroblock pair. By this measure, the basic macroblock processing structure is kept intact, and motion compensation areas are permitted to be as large as the size of a macroblock.

[0082] Each macroblock of a field macroblock pair is processed very similarly to a macroblock within a field in PAFF coding. However, since a mixture of field and frame macroblock pairs may occur within an MBAFF frame, some stages of the pre-coding procedure in precoder **12**, such as the prediction of motion vectors, the prediction of intra prediction modes, intra frame sample prediction, deblocking filtering and context modeling in entropy coding and the zig-zag scanning of transform coefficients are modified when compared to the PAFF coding in order to account for this mixture.

[0083] To summarize, the pre-coded video signal output by precoder **12** depends on the type of coding precoder **12** has decided to use. In case of MBAFF coding, as it is assumed herein, the pre-coded video signal contains a flag mb_field_decoding_flag for each non-skipped macroblock pair. The flag mb_field_decoding_flag indicates for each macroblock pair it belongs to whether the corresponding macroblocks are coded in frame or field coding mode. On decoder side, this flag is necessary in order to correctly decode the precoded video signal. In case, the macroblocks of a macroblock pair are coded in frame mode, the flag mb_field_decoding_flag is zero, whereas the flag is one in the other case.

[0084] Now, while the general mode of operation of the original decoder arrangement of FIG. **2** has been described without referring to a special bin, with respect to FIG. **5**, the functionality of this arrangement is now described with respect to the binary arithmetic coding of the bin strings of exemplary syntax elements for which the spatial relationship between the syntax element of neighboring blocks is used while MBAFF coding mode is active.

[0085] The process shown in FIG. **5** starts at the arrival of a bin value of a syntax element at the input of context modeler **112**. That is, eventually, the syntax element had to be binarized in binarizer **102** if needed, i.e. unless the syntax element is already a binary value. In a first step **300**, context modeler **112** determines as to whether the incoming bin is a bin dedicated to a context assignment based on neighboring syntax elements, i.e. syntax elements in neighboring blocks. It is recalled that the description of FIG. **5** assumes that MBAFF coding is active. If the determination in step **300** results in the incoming bin not being dedicated to context assignment based on neighboring syntax elements, another syntax element handling is performed in step **304**. In the second case, context modeler **112** determines a neighboring block of the current block to which the syntax element of the incoming bin relates. The determination process of step **306** is described in more detail below with respect to exemplary syntax elements and their bins, respectively. In any case, the determination in step **306** depends on the current macroblock to which the syntax element of the current bin relates being frame or field coded, as long as the neighboring block in question is external to the macroblock containing the current block.

[0086] Next, in step **308**, the context modeler **112** assigns a context model to the bin based on a predetermined attribute of the neighboring block. The step of assigning **308** results in a context index ctxIdx pointing to the respective entry in a table assigning each context index a probability model, to be used for binary arithmetic coding of the current bin of the current syntax element.

[0087] After the determination of ctxIdx, context modeler **112** passes the variable ctxIdx or the probability estimation status indexed by ctxIdx along with the current bin itself to regular coding engine **118**. Based on these inputs, the regular coding engine **118** arithmetically encodes, in step **322**, the bin into the bit stream **124** by using the current probability state of the context model as indexed by ctxIdx.

[0088] Thereafter, regular coding engine **118** passes the bin value via path **117** back to context modeler **112**, whereupon context modeler **112** adapts, in step **324**, the context model indexed by ctxIdx with respect to its probability estimation state. Thereafter, the process of coding the syntax element into the bit stream at the output **124** ends at **326**.

[0089] It is emphasized that the bin string into which the syntax element may be binarized before step **310** may be composed of both, bins that are arithmetically encoded by use of the current probability state of context model ctxIdx in step **322** and bins arithmetically encoded in bypass coding engine **120** by use of an equi-probable probability estimation although this is not shown in FIG. **5**. Rather, FIG. **5** merely concerns the exemplary encoding of one bin of a syntax element.

[0090] The steps **322** and **324**, encompassed by dotted line **327** in FIG. **5**, are explained in more detail below with respect to FIG. **6**.

[0091] FIG. **6** shows, on the left hand side, a flow diagram of the process **327**. On the right hand side, FIG. **6** shows a

memory **328** to which both, the context modeler **112** and the regular coding engine **118**, have access in order to load, write, and update specific variables. These variables comprise R and L, which define the current state or current probability interval of the binary arithmetical coder **100***c*. In particular, R denotes the current interval range R, while L denotes the base or lower end point of current probability interval. Thus, the current interval of the binary arithmetic coder **100***c* extends from L to L+R.

[0092] Furthermore, memory **328** contains a table **329**, which associates each possible value of ctxIdx, e.g. 0-398, a pair of a probability state index σ and an MPS value ω, both defining the current probability estimation state of the respective context model indexed by the respective context index ctxIdx. The probability state σ is an index that uniquely identifies one of a set of possible probability values p_σ. The probability values p_σ are an estimation for the probability of the next bin of that context model to be a least probable symbol (LPS). Which of the possible bin values, i.e., a null or one, is meant by the LPS, is indicated by the value of MPS ω. If ω is 1, LPS is 0 and vice-versa. Thus, the state index and MPS together uniquely define the actual probability state or probability estimation of the respective context model. Both variables divide the actual interval L to L+R into two sub-intervals, namely the first sub-interval extending from L to $L+R\cdot(1-p_\sigma)$ and the second interval extending from $L+R\cdot p_\sigma$ to L+R. The first or lower sub-interval corresponds to the most probable symbol whereas the upper sub-interval corresponds to the least probable symbol. Exemplary values for p_σ are derivable from the following recursive equation, with α being a value between about 0.8 to 0.99, and preferably being $\alpha=(0.01875/0.5)^{1/63}$, σ being an integer from 1 to 63: $p_\sigma=\alpha p_{\sigma-1}$, for all $\sigma=1, \ldots, 63$, and $p_0=0.5$.

[0093] Now, in a first step **330**, the range R_{LPS} of the lower sub-interval is determined based on R and the probability state corresponding to the chosen context model indexed by ctxIdx, later on called simply σ_i, with i being equal to ctxIdx. The determination in step **330** may comprise a multiplication of R with $p_{\sigma i}$. Nevertheless, in accordance with an alternative embodiment, the determination in step **330** could be conducted by use of a table, which assigns to each possible pair of probability state index σ_i and a variable ρ a value for R_{LPS}, such a table being shown at **332**. The variable ρ would be a measure for the value of R in some coarser units then a current resolution by which R is computationally represented.

[0094] After having determined R_{LPS}, in step **334**, regular coding engine **118** amends R to be $R-R_{LPS}$, i.e., to be the range of the lower sub-interval.

[0095] Thereafter, in step **336**, the regular coding engine **118** checks as to whether the value of the actual bin, i.e. either the already binary syntax element or one bin of a bin string obtained from the current syntax element, is equal to the most probable symbol as indicated by ω_i or not. If the current bin is the MPS, L needs not to be updated and the process transitions to step **338**, where context modeler **112** updates the probability estimation state of the current context model by updating σ_i. In particular, context modeler **112** uses a table **340** which associates each probability state index σ with an updated probability state index in case the actual symbol or bin was the most probable symbol, i.e., σ becomes transIdxMPS (σ_i).

[0096] After step **338**, the process ends at **340** where bits or a bit are added to the bit stream if possible. To be more specific, a bit or bits are added to the bit stream in order to indicate a probability value falling into the current interval as defined by R and L. In particular, step **340** is performed such that at the end of a portion of the arithmetic coding of a precoded video signal, such as the end of a slice, the bit stream defines a codeword defining a value that falls into the interval [L,L+R), thereby uniquely identifying to the decoder the bin values having been encoded into the codeword. Preferably, the codeword defines the value within the current interval having the shortest bit length. As to whether a bit or bits are added to the bit stream in step **340** or not, depends on the fact as to whether the value indicated by the bit stream will remain constant even if the actual interval is further sub-divided with respect to subsequent bins, i.e. as to whether the respective bit of the representation of the value falling in the current interval does not change whatever subdivisions will come. Furthermore, renormalization is performed in step **340**, in order to keep R and L represent the current interval within a predetermined range of values.

[0097] If in step **336** it is determined that the current bin is the least probable symbol LPS, the regular coding engine **118** actualizes the current encoder state R and L in step **342** by amending L to be L+R and R to be R_{LPS}. Then, if σ_i is equal to 0, i.e. if the probability state index indicates equal probability for both, 1 and 0, in step **344**, the value MPS is updated by computing $\omega_i=1-\omega_i$. Thereafter, in step **346**, the probability state index is actualized by use of table **340**, which also associates each current probability state index with an updated probability state index in case the actual bin value is the least probable symbol, i.e., amending σ_i to become transIdxLPS (σ_i). After the probability state index σ_i and ω_i has been adapted in steps **344** and **346**, the process steps to step **340** which has already been described.

[0098] After having described the encoding process of syntax elements by exploiting the spatial relationship between syntax element of neighboring blocks for context model assignment, the context model assignment and the definition of the neighborhood between a current and a neighboring block is described in more detail below with respect to the following syntax elements contained in the precoded video signal as output by precoder **12**. These syntax elements are listed below.

Name of the syntax element	Meaning of the syntax element
Mb_skip_flag	This flag relates to a certain macroblock of a certain slice of a video frame. Mb_skip_flag equal to 1 specifies that the current macroblock is to be skipped when performing a decoding process on the precoded video signal. Mb_skip_flag equal to 0 specifies that the current macroblock is not skipped. In particular, in the H.264/AVC standard, Mb_skip_flag equal to 1 specifies that for the current macroblock, when decoding a P or SP slice, Mb_type is inferred to be p_skip and the macroblock type is collectively referred to as P macroblock type, and when decoding a B slice, Mb_type is inferred to be B_skip and the macroblock type is collectively referred to as B macroblock type.
Mb_field_decoding_flag	Mb_field_decoding_flag equal to 0 specifies that the current macroblock pair is a frame macroblock pair and Mb_field_decoding_flag equal to 0 specifies that the macroblock pair is a field macroblock pair. Both macroblocks of a frame

-continued

Name of the syntax element	Meaning of the syntax element
	macroblock pair are referred to in the present description as frame macroblocks, whereas both macroblocks of a field macroblock pair are referred to in this text as field macroblocks.
Mb_type	Mb_type specifies the macroblock type. For example, the semantics of Mb_type in the H.264/AVC standard depends on the slice type. Depending on the slice type, Mb_type can assume values in the range of 0 to 25, 0 to 30, 0 to 48 or 0-26, depending on the slice type.
Coded_block_pattern	Coded_block_pattern specifies which of a sub-part of the current macroblock contains non-zero transform coefficients. Transform coefficients are the scalar quantities, considered to be in a frequency domain, that are associated with a particular one-dimensional or two-dimensional frequency index in an inverse transform part of the decoding process. To be more specific, each macroblock 10a - irrespective of the macroblock being a frame coded macroblock (FIG. 4a) or a field coded macroblock (FIG. 4b), is partitioned into smaller sub-parts, the sub-parts being arrays of size 8 × 8 pixel samples. Briefly referring to FIG. 4a, the pixels 1 to 8, 11 to 18, 21 to 28, . . . , 71 to 78 could form the upper left block of luma pixel samples in the top macroblock 250 of macroblock pair 10b. This top macroblock 250 would comprise another three of such blocks, all four blocks arranged in a 2 × 2 array. The same applies for the bottom macroblock 252 and also applies for field coded macroblocks as shown in FIG. 4b, where, for example, pixels 1 to 8, 21 to 28, 41 to 48, . . . , 141 to 148 would form the upper left block of the top macroblock. Thus, for each macroblock coded, the precoded video signal output by precoder 12 would comprise one or several syntax elements coded_block_pattern. The transformation from spatial domain to frequency domain, could be performed on these 8 × 8 sub-parts or on some smaller units, for example, 4 × 4 sub-arrays, wherein each 8 × 8 sub-part comprises 4 smaller 4 × 4 partitions. The present description mainly concerns luma pixel samples. Nevertheless, the same could also apply accordingly for chroma pixel samples.
ref_Idx_l0/ ref_Idx_l1	This syntax element concerns the prediction of the pixel samples of a macroblock during encoding and decoding. In particular, ref_Idx_l0, when present in the precoded video signal output by precoder 12, specifies an index in a list 0 of a reference picture to be used for prediction. The same applies for ref_Idx_l1 but with respect to another list of the reference picture.
mvd_l0/mvd_l1	mvd_l0 specifies the difference between a vector component to be used for motion prediction and the prediction of the vector component. The same applies for mvd_l1, the only difference being, that same are applied to different reference picture lists. ref_Idx_l0, ref_Idx_l1, mvd_l0 and mvd_l1 all relate to a particular macroblock partition. The partitioning of the macroblock is specified by Mb_type.
intra_chroma_pred_mode	Intra_chroma_pred_mode specifies the type of spatial prediction used for chroma whenever any part of the luma macroblock is intra-coded. In intra prediction, a prediction is derived from the decoded samples of the same decoded picture or frame. Intra prediction is contrary to inter prediction where a prediction is derived from decoded samples of reference pictures other than the current decoded picture.

-continued

Name of the syntax element	Meaning of the syntax element
coded_block_flag	coded_block_flag relates to blocks of the size of 4 × 4 picture samples. If coded_block_flag is equal to 0, the block contains no non-zero transform coefficients. If coded_block_flag is equal to 1, the block contains at least one non-zero transform coefficient.

[0099] As can be gathered from the above table, some of these syntax elements relate to a current macroblock in the whole, whereas others relate to sub-parts, i.e., sub-macroblocks or partitions thereof, of a current macroblock. In a similar way, the assignment of a context model to these syntax elements is dependent on syntax elements of either neighboring macroblocks, neighboring sub-macroblocks or neighboring partitions thereof. FIG. **9** illustrates the partition of macroblocks (upper row) and sub-macroblocks (lower row). The partitions are scanned for inter prediction as shown in FIG. **9**. The outer rectangles in FIG. **9** refer to the samples in a macroblock or sub-macroblock, respectively. The inner rectangles refer to the partitions. The number in each inner rectangle specifies the index of the inverse macroblock partition scan or inverse sub-macroblock partition scan.

[0100] Before describing in detail the dependency of the context model assignment on the syntax element of neighboring blocks, with respect to FIG. **7**, it is described, how the addresses of the top macroblock of the macroblock pair to the left and above the current macroblock pair may be computed, since these are the possible candidates, which comprise the syntax element in the block to the left of and above the current block containing the current syntax element to be arithmetically encoded. In order to illustrate the spatial relationships, in FIG. **7**, a portion of six macroblock pairs of a video frame is shown, wherein each rectangle region in FIG. **7** corresponds to one macroblock and the first and the second two vertically adjacent macroblocks in each column form a macroblock pair.

[0101] In FIG. **7**, CurrMbAddr denotes the macroblock address of the top macroblock of the current macroblock pair, the current syntax element is associated with or relates to. The current macroblock pair is encompassed by bold lines. In other words, they from the border of a macroblock pair. mbAddrA and mbAddrB denote the addresses of the top macroblocks of the macroblock pairs to the left and above the current macroblock pair, respectively.

[0102] In order to compute the addresses of the top macroblock of the neighboring macroblock pair to the left and above the current macroblock pair, context modeler **112** computes

$$MbAddrA=2\cdot(CurrMbAddr/2-1)$$

$$MbAddrB=2\cdot(CurrMbAddr/2-PicWidthInMbs)$$

where PicWidthInMbs specifies the picture within units of macroblocks. The equations given above can be understood by looking at FIG. **3**. It is noted that in FIG. **3** the picture width in units of macroblocks has been denoted i. It is further noted that the equations given above are also true when the current macroblock address CurrMbAddress is interchanged with the odd numbered macroblock address of the bottom macroblock of the current macroblock pair, i.e., CurrMbAddress+1,

because in the equation above, "/" denotes an integer division with transaction of the result towards zero. For example, 7/4 and −7/−4 are truncated to 1 and −7/4 and 7/−1 are truncated to −1.

[0103] Now, after having described how to compute neighboring macroblocks, it is briefly recalled that each macroblock contains 16×16 luma samples. These luma samples are divided up into four 8×8 luma blocks. These luma blocks may be further subdivided into 4×4 luma blocks. Furthermore, for the following description, each macroblock further comprises 8×8 luma samples, i.e., the pixel width of the chroma samples being doubled compared to luma samples. These 8×8 chroma samples of a macroblock are divided up into four 4×4 luma blocks. The blocks of a macroblock are numbered. Accordingly, the four 8×8 luma blocks each have a respective block address uniquely indicating each 8×8 block in the macroblock. Next, each pixel sample in a macroblock belongs to a position (x, y) wherein (x, y) denotes the luma or chroma location of the upper-left sample of the current block in relation to the upper-left luma or chroma sample of the macroblock. For example, with respect to luma samples, the pixel **23** in top macroblock **252** in FIG. **4***b* would have the pixel position (2, 1), i.e., third column, second row.

[0104] After having described this, the derivation process of ctxIdx for at least some of the bins of syntax elements listed in the above table is described.

[0105] With respect to the syntax element mb_skip_flag, the context modeler assignment depends on syntax elements relating to neighboring macroblocks. Thus, in order to determine the context index ctxIdx the addresses mbAddrA and mbAddrB are determined as described above. Then, let condTermN (with N being either A or B) be a variable that is set as follows:

[0106] If mbAddrN is not available or mb_skip_flag for the macroblock mbAddrN is equal to 1, conTermN is set to 0

[0107] Otherwise, condTermN is set to 1.

[0108] ctxIdx is derived based on an context index incrementor ctxIdxInc=conTermA+condTermB.

[0109] For the syntax element mb_field_decoding_flag, ctxIdx is determined as follows:

[0110] Let condTermN (with N being either A or B) be a variable that is set as follows.

[0111] If any of the following conditions is true, then condTermN is set to 0,

[0112] mbAddrN is not available

[0113] the macroblock mbAddrN is a frame macroblock.

[0114] Otherwise, condTermN is set to 1.

[0115] ctxIdx is derived based on an context index incrementor ctxIdxInc=condTermA+condTermB

[0116] wherein mbaddrN is not available, if

[0117] (((CurrMbAddr/2)%(PicWidthInMbs))==0).

[0118] For the syntax element Mb_type, ctxIdx is determined dependent on the availability of macroblocks mbAddrN (with N being either A or B), and the syntax element Mb_type of this neighboring macroblocks.

[0119] With respect to the other syntax element listed in the above table, the dependency of the context modeler assignment is defined accordingly, wherein for syntax elements, which relate to blocks smaller than a macroblock, the assignment is also dependent on syntax element relating to such smaller blocks being smaller than macroblocks. For example, for the syntax element coded_block_pattern, the context index assignment is dependent not only on the availability of macroblock MbAddrN and the syntax element Mb_type of the macroblock MbAddrN but also on the syntax element Coded_block_pattern of the neighboring block. Further, it is worth noting that the syntax elements listed above are all dependent on the respective syntax element of the neighboring block. Differing thereto, the context model assignment of syntax elements mvd_l0, mvd_l1, ref_idx_l0 and ref_idx_l1 is not dependent on the respective syntax elements of the neighboring block. The context modeler assignment for intra_chroma_pred_mode is dependent on mbAddrN availability, macroblock mbAddrN being coded in inter prediction mode or not, Mb_type for the macroblock mbAddrN and the syntax element intra_chroma_pred_mode for the macroblock MbAddrN. The syntax element coded_block_flag context model assignment is dependent on the availability of MbAddrN, the current macroblock being coded in inter prediction mode, Mb_type for the macroblock mbAddrN and the syntax element coded_block_flag of the neighboring block.

[0120] In the following, it is described, how a neighboring block is determined. In particular, this involves computing mbAddrN and the block index indexing the sub-part of the macroblock MbAddrN, this sub-part being the neighboring block of the current block.

[0121] The neighborhood for slices using macroblock adaptive frames/field coding as described in the following in accordance with an embodiment of the present invention is defined in a way that guarantees that the areas covered by neighboring blocks used for context modeling in context adaptive binary arithmetic coding inside an MBAFF-frame adjoin to the area covered by the current block. This generally improves the coding efficiency of a context adaptive arithmetic coding scheme as it is used here in connection with the coding of MBAFF-slices in comparison to considering each macroblock pair as frame macroblock pair for the purpose of context modeling as described in the introductory portion of the specification, since the conditional probabilities estimated during the coding process are more reliable.

[0122] The general concept of defining the neighborhood between a current and a reference block is described in the following section 1.1. In section 1.2, a detailed description, which specifies how the neighboring blocks, macroblocks, or partitions to the left of and above the current block, macroblock, or partition are obtained for the purpose of context modeling in context adaptive binary arithmetic coding, is given.

1.1. General Concept Neighborhood Definition

[0123] Let (x, y) denote the luma or chroma location of the upper-left sample of the current block in relation to the upper-left luma or chroma sample of the picture CurrPic. The variable CurrPic specifies the current frame, which is obtained by interleaving the top and the bottom field, if the current block is part of a macroblock pair coded in frame mode (mb_field_decoding_flag is equal to 0). If the current block is or is part of a top field macroblock, CurrPic specifies the top field of the current frame; and if the current block is or is part of a bottom field macroblock, CurrPic specifies the bottom field of the current frame.

[0124] Let (xA, yA) and (xB, yB) denote the luma or chroma location to the left of and above the location (x, y), respectively, inside the picture CurrPic. The locations (xA, yA) and (xB, yB) are specified by

$$(xA,yA)=(x-1,y)$$

$$(xB,yB)=(x,y-1)$$

[0125] The block to the left of the current block is defined as the block that contains the luma or chroma sample at location (xA, yA) relative to the upper-left luma or chroma sample of the picture CurrPic and the block above the current block is defined as the block that contains the luma or chroma sample at location (xB, yB) relative to the upper-left luma or chroma sample of the picture CurrPic. If (xA, yA) or (xB, yB) specify a location outside the current slice, the corresponding block is marked as not available.

1.2. Detailed Description of Neighborhood Definition

[0126] The algorithm described in Sec. 1.2.1 specifies a general concept for MBAFF-slices that describes how a luma sample location expressed in relation to the upper-left luma sample of the current macroblock is mapped onto a macroblock address, which specifies the macroblock that covers the corresponding luma sample, and a luma sample location expressed in relation to the upper-left luma sample of that macroblock. This concept is used in the following Sec. 1.2.2-1.2.6.

[0127] The Sec. 1.2.2-1.2.6 describe how the neighboring macroblocks, 8×8 luma blocks, 4×4 luma blocks, 4×4 chroma block, and partitions to the left of and above a current macroblock, 8×8 luma block, 4×4 luma block, 4×4 chroma block, or partition are specified. These neighboring macroblock, block, or partitions are needed for the context modeling of CABAC for the following syntax elements: mb_skip_flag, mb_type, coded_block_pattern, intra_chroma_pred_mode, coded_block_flag, ref_idx_l0, ref_idx_l1, mvd_l0, mvd_l1.

1.2.1 Specification of Neighboring Sample Locations

[0128] Let (xN, yN) denote a given luma sample location expressed in relation to the upper-left luma sample of the current macroblock with the macroblock address CurrMbAddr. It is recalled that in accordance with the present embodiment each macroblock comprises 16×16 luma samples. xN and yN lie within −1 . . . 16. Let mbAddrN be the macroblock address of the macroblock that contains (xN, yN), and let (xW,yW) be the, location (xN, yN) expressed in relation to the upper-left luma sample of the macroblock mbAddrN (rather than relative to the upper-left luma sample of the current macroblock).

[0129] Let mbAddrA and mbAddrB specify the macroblock address of the top macroblock of the macroblock pair to the left of the current macroblock pair and the top macroblock of the macroblock pair above the current macroblock pair, respectively. Let PicWidthInMbs be a variable that specifies the picture width in units of macroblocks. mbAddrA and mbAddrB are specified as follows.

$$mbAddrA=2*(CurrMbAddr/2-1)$$

[0130] If mbAddrA is less than 0, or if (CurrMbAddr/2) % PicWidthInMbs is equal to 0, or if the macroblock with address mbAddrA belongs to a different slice than the current slice, mbAddrA is marked as not available.

$$mbAddrB=2*(CurrMbAddr/2-PicWidthInMbs)$$

[0131] If mbAddrB is less than 0, or if the macroblock with address mbAddrB belongs to a different slice than the current slice, mbAddrB is marked as not available.

[0132] The Table in FIG. **8** specifies the macroblock address mbAddrN and a variable yM in the following two ordered steps:

[0133] 1. Specification of a macroblock address mbAddrX (fifth column) depending on (xN, yN) (first and second column) and the following variables:

[0134] The variable currMbFrameFlag (third column) is set to 1, if the current macroblock with address CurrMbAddr is a part of a frame macroblock pair; otherwise it is set to 0.

[0135] The variable mb1sTopMbFlag (forth column) is set to 1, if CurrMbAddr%2 is equal to 0; otherwise it is set to 0.

[0136] 2. Depending on the availability of mbAddrX (fifth column), the following applies:

[0137] If mbAddrX (which can be either mbAddrA or mbAddrB) is marked as not available, mbAddrN is marked as not available.

[0138] Otherwise (mbAddrX is available), mbAddrN is marked as available and Table 1 specifies mbAddrN and yM depending on (xN, yN) (first and second column), currMbFrameFlag (third column), mblsTopMbFlag (forth column), and the variable mbAddrXFrameFlag (sixth column), which is derived as follows:

[0139] mbAddrXFrameFlag is set to 1, if the macroblock mbAddrX is a frame macroblock; otherwise it is set to 0.

[0140] Unspecified values of the above flags in Table 1 indicate that the value of the corresponding flags is not relevant for the current table rows.

[0141] To summarize: in the first four columns, the input values xN, yN, currMbFrameFlag and MblsTopMbFlag are entered. In particular, the possible input values for parameters xN and yN are −1 to 16, inclusive. These parameters determine mbAddrX listed in the fifth column, i.e. the macroblock pair containing the wanted luma sample. The next two columns, i.e., the sixth and the seventh column, are needed to obtain the final output mbAddrN and yN. These further input parameters are MbAddrXFrameFlag indicating as to whether a macroblock pair indicated by mbAddrX is frame or field coded, and some additional conditions concerning as to whether yN is even or odd numbered or is greater than or equal to 8 or not.

[0142] As can be seen, when xN and yN are both positive or zero, i.e., the wanted pixel sample lies within the current macroblock relative to which xN and yN are defined, the output macroblock address does not change, i.e., it is equal to CurrMbAddr. Moreover, yM is equal yM. This changes when the input xM and yM indicates a pixel sample lying outside the current macroblock, i.e., to the left (xN<0) all to the top of the current macroblock (yN<0).

[0143] Outgoing from the result of the table of FIG. **8**, the neighboring luma location (xW, yW) relative to the upper-left luma sample of the macroblock-mbAddrN is specified as

$$xW=(xN+16)\%16$$

$$yW=(yM+16)\%16.$$

[0144] It is emphasized that the aforementioned considerations pertained for illustrative purposes merely luma samples. The considerations are slightly different when considering chroma samples since a macroblock contains merely 8×8 chroma samples.

1.2.2 Specification of Neighboring Macroblocks

[0145] The specification of the neighboring macroblocks to the left of and above the current macroblock is used for the

context modeling of CABAC for the following syntax elements: mb_skip_flag, mb_type, coded_block_pattern, intra_chroma_prediction mode, and coded_block_flag.

[0146] Let mbAddrA be the macroblock address of the macroblock to the left of the current macroblock, and mbAddrB be the macroblock address of the macroblock above the current macroblock.

[0147] mbAddrA, mbAddrB, and their availability statuses are obtained as follows:

[0148] mbAddrA and its availability status are obtained as described in Sec. 1.2.1 given the luma location (xN, yN)=(−1, 0).

[0149] mbAddrB and its availability status are obtained as described in Sec. 1.2.1 given the luma location (xN, yN)=(0, −1).

1.2.3 Specification of Neighboring 8×8 Luma Blocks

[0150] The specification of the neighboring 8×8 luma blocks to the left of and above the current 8×8 luma block is used for the context modeling of CABAC for the syntax element coded_block_pattern.

[0151] Let luma8×8BlkIdx be the index of the current 8×8 luma block inside the current macroblock CurrMbAddr. An embodiment of the assignment of block index luma8×8BlkIdx to the respective blocks within a macroblock is shown in FIG. **9** (upper-right corner).

[0152] Let mbAddrA be the macroblock address of the macroblock that contains the 8×8 luma block to the left of the current 8×8 luma block, and let mbAddrB be the macroblock address of the macroblock that contains the 8×8 luma block above the current 8×8 luma block. Further, let luma8×8BlkIdxA be the 8×8 luma block index (inside the macroblock mbAddrA) of the 8×8 luma block to the left of the current 8×8 luma block, and let luma8×8BlkIdXB be the 8×8 luma block index (inside the macroblock mbAddrB) of the 8×8 luma block above the current 8×8 luma block.

[0153] mbAddrA, mbAddrB, luma8×8BlkIdxA, luma8×8BlkIdxB, and their availability statuses are obtained as follows:

[0154] Let (xC, yC) be the luma location of the upper-left sample of the current 8×8 luma block relative to the upper-left luma sample of the current macroblock.

[0155] mbAddrA, its availability status, and the luma location (xW, yW) are obtained as described in Sec. 1.2.1 given the luma location (xN, yN)=(xC−1, yC). If mbAddrA is available, then luma8×8BlkIdxA is set in a way that it refers to the 8×8 luma block inside the macroblock mbAddrA that covers the luma location (xW, yW); otherwise, luma8×8BlkIdA is marked as not available.

[0156] mbAddrB, its availability status, and the luma location (xW, yW) are obtained as described in Sec. 1.2.1 given the luma location (xN, yN)=(xC, yC−1). If mbAddrB is available, then luma8×8BlkIdxB is set in a way that it refers to the 8×8 luma block inside the macroblock mbAddrB that covers the luma location (xW, yW); otherwise,

[0157] luma8×8BlkIdxB is marked as not available.

1.2.4 Specification of Neighboring 4×4 Luma Blocks

[0158] The specification of the neighboring 4×4 luma blocks to the left of and above the current 4×4 luma block is used for the context modeling of CABAC for the syntax element coded_block_flag.

[0159] Let luma4×4BlkIdx be the index (in decoding order) of the current 4×4 luma block inside the current macroblock CurrMbAddr. For example, luma4×4BlkIdx could be defined as luma8×8BlkIdx of the 8×8 block containing the 4×4block multiplied by 4 plus the partition number as shown in the bottom-right corner of FIG. **9**.

[0160] Let mbAddrA be the macroblock address of the macroblock that contains the 4×4 luma block to the left of the current 4×4 luma block, and let mbAddrB be the macroblock address of the macroblock that contains the 4×4 luma block above the current 4×4 luma block. Further, let luma4×4BlkIdxA be the 4×4 luma block index (inside the macroblock mbAddrA) of the 4×4 luma block to the left of the current 4×4 luma block, and let luma4×4BlkIdxB be the 4×4 luma block index (inside the macroblock mbAddrB) of the 4×4 luma block above the current 4×4 luma block.

[0161] mbAddrA, mbAddrB, luma4×4BlkIdxA, luma4×4BlkIdxB, and their availability statuses are obtained as follows:

[0162] Let (xC, yC) be the luma location of the upper-left sample of the current 4×4 luma block relative to the upper-left luma sample of the current macroblock.

[0163] mbAddrA, its availability status, and the luma location (xW, yW) are obtained as described in Sec. 1.2.1 given the luma location (xN, yN)=(xC−1, yC). if mbAddrA is available, then luma4×4BlkIdxA is set in a way that it refers to the 4×4 luma block inside the macroblock mbAddrA that covers the luma location (xW, yW); otherwise, luma4×4BlkIdxA is marked as not available.

[0164] mbAddrB, its availability status, and the luma location (xW, yW) are obtained as described in Sec. 1.2.1 given the luma location (xN, yN)=(xC, yC−1). If mbAddrB is available, then luma4×4BlkIdxB is set in a way that it refers to the 4×4 luma block inside the macroblock mbAddrB that covers the luma location (xW, yW); otherwise, luma4×4BlkIdxB is marked as not available.

1.2.5 Specification of Neighboring 4×4 Chroma Blocks

[0165] The specification of the neighboring 4×4 chroma blocks to the left of and above the current 4×4 chroma block is used for the context modeling of CABAC for the syntax element coded_block_flag.

[0166] Let chroma4×4BlkIdx be the index (in decoding order) of the current 4×4 chroma block inside the current macroblock CurrMbAddr.

[0167] Let mbAddrA be the macroblock address of the macroblock that contains the 4×4 chroma block to the left of the current 4×4 chroma block, and let mbAddrB be the macroblock address of the macroblock that contains the 4×4 chroma block above the current 4×4 chroma block. Further, let chroma4×4BlkIdxA be the 4×4 chroma block index (inside the macroblock mbAddrA) of the 4×4 chroma block to the left of the current 4×4 chroma block, and let chroma4×4BlkIdxB be the 4×4 chroma block index (inside the macroblock mbAddrB) of the 4×4 chroma block above the current 4×4 chroma block.

[0168] mbAddrA, mbAddrB, chroma4×4BlkIdxA, chroma4×4BlkIdxB, and their availability statuses are obtained as follows:

[0169] Given luma8×8BlkIdx=chroma4×4BlkIdx, the variables mbAddrA, mbAddrB, luma8×8BlkIdxA,

luma8×8BlkIdxB, and their availability statuses are obtained as described in Sec. 1.2.3.

[0170] If luma8×8BlkIdxA is available, chroma4×4BlkIdxA is set equal to luma8×8BlkIdxA; otherwise chroma4×4BlkIdxA is marked as not available.

[0171] If luma8×8BlkIdxB is available, chroma4×4BlkIdxB is set equal to luma8×8BlkIdxB; otherwise chroma4×4BlkIdxB is marked as not available.

1.2.6 Specification of Neighboring Partitions

[0172] The specification of the neighboring partitions to the left of and above the current partition is used for the context modeling of CABAC for the following syntax elements: ref_idx_l0, ref_idx_l1, mvd_l0, mvd_l1.

[0173] Let mbPartIdx and subMbPartIdx be the macroblock partition and sub-macroblock partition indices that specify the current partition inside the current macroblock CurrMbAddr. An example for such partition indices is shown in FIG. **9**.

[0174] Let mbAddrA be the macroblock address of the macroblock that contains the partition to the left of the current partition, and let mbAddrB be the macroblock address of the macroblock that contains the partition above the current partition. Further, let mbPartIdxA and subMbPartIdxA be the macroblock partition and sub-macroblock partition indices (inside the macroblock mbAddrA) of the partition to the left of the current partition, and let mbPartIdxB and subMbPartIdxB be the macroblock partition and sub-macroblock partition indices (inside the macroblock mbAddrB) of the partition above the current partition.

[0175] mbAddrA, mbAddrB, mbPartIdxA, subMbPartIdxA, mbPartIdxB, subMbPartIdxB, and their availability statuses are obtained as follows:

[0176] Let (xC, yC) be the luma location of the upper-left sample of the current partition given by mbPartIdx and subMbPartIdx relative to the upper-left luma sample of the current macroblock.

[0177] mbAddrA, its availability status, and the luma location (xW, yW) are obtained as described in Sec. 1.2.1 given the luma location (xN,yN)=(xC−1, yC). If mbAddrA is not available, mbPartIdxA and subMbPartIdxA are marked as not available; otherwise mbPartIdxA is set in a way that it refers to the macroblock partition inside the macroblock mbAddrA that covers the luma location (xW, yW), and subMbPartIdxA is set in a way that it refers to the sub-macroblock partition inside the macroblock partition mbPartIdxA (inside the macroblock mbAddrA) that covers the luma location (xW, yW).

[0178] mbAddrB, its availability status, and the luma location (xW, yW) are obtained as described in Sec. 1.2.1 given the luma location (xN, yN)=(xC, yC−1). If mbAddrB is not available, mbPartIdxB and subMbPartIdxB are marked as not available; otherwise mbPartIdxB is set in a way that it refers to the macroblock partition inside the macroblock mbAddrB that covers the luma location (xW, yW), and subMbPartIdxB is set in a way that it refers to the sub-macroblock partition inside the macroblock partition mbPartIdxB (inside the macroblock mbAddrB) that covers the luma location (xW, yW).

[0179] After having described how to encode the above syntax elements or the bin strings or part of their bins into an arithmetically coded bit stream, the decoding of said bit stream and the retrieval of the bins is described with respect to FIGS. **10** to **12**

[0180] FIG. **10** shows a general view of a video decoder environment to which the present invention could be applied. An entropy decoder **400** receives the arithmetically coded bit stream as described above and treats it as will be described in more detail below with respect to FIGS. **11-12**. In particular, the entropy decoder **400** decodes the arithmetically coded bit stream by binary arithmetic decoding in order to obtain the precoded video signal and, in particular, syntax elements contained therein and passes same to a precode decoder **402**. The precode decoder **402** uses the syntax elements, such as motion vector components and flags, such as the above listed syntax elements, in order to retrieve, macroblock by macroblock and then slice after slice, the picture samples of pixels of the video frames **10**.

[0181] FIG. **11** now shows the decoding process performed by the entropy decoder **400** each time a bin is to be decoded. Which bin is to be decoded depends on the syntax element which is currently expected by entropy decoder **400**. This knowledge results from respective parsing regulations.

[0182] In the decoding process, first, in step **500**, the decoder **400** checks as to whether the next bin to decode is a bin of a syntax element of the type corresponding to context model assignment based on neighboring syntax elements. If this is not the case, decoder **400** proceeds to another syntax element handling in step **504**. However, if the check result in step **500** is positive, decoder **400** performs in steps **506** and **508** a determination of the neighboring block of the current block which the current bin to decode belongs to and an assignment of a context model to the bin based on a predetermined attribute of the neighboring block determined in step **506**, wherein steps **506** and **508** correspond to steps **306** and **308** of encoding process of FIG. **5**. The result of these steps is the context index ctxIdx. Accordingly, the determination of ctxIdx is performed in steps **506** and **508** in the same way as in the encoding process of FIG. **5** in steps **306** and **308** in order to determine the context model to be used in the following arithmetical decoding.

[0183] Then, in step **522**, the entropy decoder **400** arithmetically decodes the actual bin, from the arithmetically coded bit stream by use of the actual probability state of the context module as indexed by ctxIdx obtained in steps **510** to **520**. The result of this step is the value for the actual bin. Thereafter, in step **524**, the ctxIdx probability state is adapted or updated, as it was the case in step **224**. Thereafter, the process ends at step **526**.

[0184] Of course, the individual bins that are obtained by the process shown in FIG. **11** represent the syntax element value merely in case the syntax element is of a binary type. Otherwise, a step corresponding to the binarization has to be performed in reverse manner in order to obtain from the bin strings the actual value of the syntax element.

[0185] FIG. **12** shows the steps **522** and **524** being encompassed by dotted line **527** in more detail on the left hand side. On the right hand side, indicated with **564**, FIG. **11** shows a memory and its content to which entropy decoder **400** has access in order to load, store and update variables. As can be seen, entropy decoder manipulates or manages the same variables as entropy coder **14** since entropy decoder **400** emulates the encoding process as will be described in the following.

[0186] In a first step **566**, decoder **400** determines the value R_{LPS}, i.e. the range of the subinterval corresponding to the

next bin being the LPS, based on R and σ_i. Thus, step **566** is identical to step **330**. Then, in step **568**, decoder **400** computes $R_{MPS}=R-R_{LPS}$ with R_{MPS} being the range of the subinterval associated with the most probable symbol. The current interval from L to R is thus subdivided into subintervals L to $L+R_{MPS}$ and $L+R_{MPS}$ to L+R. Now, in step **570** decoder **400** checks as to whether the value of the arithmetic coding codeword in the arithmetically coded bit stream falls into the lower or upper subinterval. The decoder **400** knows that the actual symbol bin, is the most probable symbol as indicated by ω_i when the value of the arithmetic codeword falls into the lower subinterval and accordingly sets the bin value to the value of ω_i in step **572**. In case the value falls into the upper subinterval, decoder **400** sets the symbol to be $1-\omega_i$ in step **574**. After step **572**, the decoder **400** actualizes the decoder state or the current interval as defined by R and L by setting R to be R_{MPS} in step **574**. Then, in step **576**, the decoder **400** adapts or updates the probability state of the current context model i as defined by σ_i and ω_i by transitioning the probability state index σ_i as was described with respect to step **338** in FIG. **9**. Thereafter, the process **527** ends at step **578**.

[0187] After step **574**, the decoder actualizes the decoder state in step **580** by computing L=L+R and $R=R_{LPS}$. Thereafter, the decoder **400** adapts or updates the probability state in steps **582** and **584** by computing $\omega_i=1-\omega_i$ in step **582**, if σ_i is equal to 0, and transitioning the probability state index σ_i to a new probability state index in the same way as described with respect to step **346** in FIG. **9**. Thereafter, the process ends at step **578**.

[0188] After having described the present invention with respect to the specific embodiments, it is noted that the present invention is not restricted to these embodiments. In particular, the present invention is not restricted to the specific examples of syntax elements. Moreover, the assignment in accordance with steps **308** and **408** does not have to be dependent on syntax elements of neighboring blocks, i.e., syntax elements contained in the precoded video signal output by precoder **12**. Rather, the assignment may be dependent on other attributes of the neighboring blocks. Moreover, the definition of neighborhoods between neighboring blocks is described with respect to the table of FIG. **8** may be varied. Further, the pixel samples of the two interlaced fields could be arranged in another way than described above.

[0189] Moreover, other block sizes than 4×4 blocks could be used as a basis for the transformation, and, although in the above embodiment the transformation was applied to picture sample differences to a prediction, the transformation could be as well applied to the picture sample itself without performing a prediction. Furthermore, the type of transformation is not critical. DCT could be used as well as a FFT or wavelet transformation. Furthermore, the present invention is not restricted to binary arithmetic encoding/decoding. The present invention can be applied to multi-symbol arithmetic encoding as well. Additionally, the sub-divisions of the video frame into slices, macroblock pairs, macroblocks, picture elements etc. was for illustrating purposes only, and this is not to restrict the scope of the invention to this special case.

[0190] In the following, reference is made to FIG. **13** to show, in more detail than in FIG. **1**, the complete setup of a video encoder engine including an entropy-encoder as it is shown in FIG. **13** in block **800** in which the aforementioned arithmetic coding of syntax elements by use of a context assignment based on neighboring syntax elements is used. In particular, FIG. **13** shows the basic coding structure for the emerging H.264/AVC standard for a macroblock. The input video signal is, split into macroblocks, each macroblock having 16×16 luma pixels. Then, the association of macroblocks to slice groups and slices is selected, and, then, each macroblock of each slice is processed by the network of operating blocks in FIG. **13**. It is to be noted here that an efficient parallel processing of macroblocks is possible, when there are various slices in the picture. The association of macroblocks to slice groups and slices is performed by means of a block called coder control **802** in FIG. **13**. There exist several slices, which are defined as follows:

[0191] I slice: A slice in which all macroblocks of the slice are coded using intra prediction.

[0192] P slice: In addition, to the coding types of the I slice, some macroblocks of the P slice can also be coded using inter prediction with at most one motion-compensated prediction signal per prediction block.

[0193] B slice: In addition, to the coding types available in a P slice, some macroblocks of the B slice can also be coded using inter prediction with two motion-compensated prediction signals per prediction block.

[0194] The above three coding types are very similar to those in previous standards with the exception of the use of reference pictures as described below. The following two coding types for slices are new:

[0195] SP slice: A so-called switching P slice that is coded such that efficient switching between different precoded pictures becomes possible.

[0196] SI slice: A so-called switching I slice that allows an exact match of a macroblock in an SP slice for random access and error recovery purposes.

[0197] Slices are a sequence of macroblocks, which are processed in the order of a raster scan when not using flexible macroblock ordering (FMO). A picture maybe split into one or several slices as shown in FIG. **15**. A picture is therefore a collection of one or more slices. Slices are self-contained in the sense that given the active sequence and picture parameter sets, their syntax elements can be parsed from the bit stream and the values of the samples in the area of the picture that the slice represents can be correctly decoded without use of data from other slices provided that utilized reference pictures are identical at encoder and decoder. Some information from other slices maybe needed to apply the deblocking filter across slice boundaries.

[0198] FMO modifies the way how pictures are partitioned into slices and macroblocks by utilizing the concept of slice groups. Each slice group is a set of macroblocks defined by a macroblock to slice group map, which is specified by the content of the picture parameter set and some information from slice headers. The macroblock to slice group map consists of a slice group identification number for each macroblock in the picture, specifying which slice group the associated macroblock belongs to. Each slice group can be partitioned into one or more slices, such that a slice is a sequence of macroblocks within the same slice group that is processed in the order of a raster scan within the set of macroblocks of a particular slice group. (The case when FMO is not in use can be viewed as the simple special case of FMO in which the whole picture consists of a single slice group).

[0199] Using FMO, a picture can be split into many macroblock-scanning patterns such as interleaved slices, a dispersed macroblock allocation, one or more "foreground" slice groups and a "leftover" slice group, or a checker-board type of mapping.

[0200] Each macroblock can be transmitted in one of several coding types depending on the slice-coding type. In all slice-coding types, the following types of intra coding are supported, which are denoted as Intra_4×4 or Intra_16×16 together with chroma prediction and I_PCM prediction modes.

[0201] The Intra_4×4 mode is based on predicting each 4×4 luma block separately and is well suited for coding of parts of a picture with significant detail. The Intra_16×16 mode, on the other hand, does prediction of the whole 16×16 luma block and is more suited for coding very smooth areas of a picture.

[0202] In addition, to these two types of luma prediction, a separate chroma prediction is conducted. As an alternative to Intra 4×4 and Intra_16×16, the I_PCM coding type allows the encoder to simply bypass the prediction and transform coding processes and instead directly send the values of the encoded samples. The I_PCM mode serves the following purposes:

[0203] 1. It allows the encoder to precisely represent the values of the samples

[0204] 2. It provides a way to accurately represent the values of anomalous picture content without significant data expansion

[0205] 3. It enables placing a hard limit on the number of bits a decoder must handle for a macroblock without harm to coding efficiency.

[0206] In contrast to some previous video coding standards (namely H.263+ and MPEG-4 Visual), where intra prediction has been conducted in the transform domain, intra prediction in H.264/AVC is always conducted in the spatial domain, by referring to the bins of neighboring samples of previously coded blocks which are to the left and/or above the block to be predicted. This may incur error propagation in environments with transmission errors that propagate due to motion compensation into inter-coded macroblocks. Therefore, a constrained intra coding mode can be signaled that allows prediction only from intra-coded neighboring macroblocks.

[0207] When using the Intra_4×4 mode, each 4×4 block is predicted from spatially neighboring samples as illustrated on the left-hand side of FIG. **16**. The 16 samples of the 4×4 block, which are labeled as a-p, are predicted using prior decoded samples in adjacent blocks labeled as A-Q. For each 4×4 block one of nine prediction modes can be utilized. In addition, to "DC" prediction (where one value is used to predict the entire 4×4 block), eight directional prediction modes are specified as illustrated on the right-hand side of FIG. **14**. Those modes are suitable to predict directional structures in a picture such as edges at various angles.

[0208] In addition, to the intra macroblock coding types, various predictive or motion-compensated coding types are specified as P macroblock types. Each P macroblock type corresponds to a specific partition of the macroblock into the block shapes used for motion-compensated prediction. Partitions with luma block sizes of 16×16, 16×8, 8×16, and 8×8 samples are supported by the syntax. In case partitions with 8×8 samples are chosen, one additional syntax element for each 8×8 partition is transmitted. This syntax element specifies whether the corresponding 8×8 partition is further partitioned into partitions of 8×4, 4×8, or 4×4 luma samples and corresponding chroma samples.

[0209] The prediction signal for each predictive-coded M×N luma block is obtained by displacing an area of the corresponding reference picture, which is specified by a translational motion vector and a picture reference index. Thus, if the macroblock is coded using four 8×8 partitions and each 8×8 partition is further split into four 4×4 partitions, a maximum of sixteen motion vectors may be transmitted for a single P macroblock.

[0210] The quantization parameter SliceQP is used for determining the quantization of transform coefficients in H.264/AVC. The parameter can take 52 values. Theses values are arranged so that an increase of 1 in quantization parameter means an increase of quantization step size by approximately 12% (an increase of 6 means an increase of quantization step size by exactly a factor of 2). It can be noticed that a change of step size by approximately 12% also means roughly a reduction of bit rate by approximately 12%.

[0211] The quantized transform coefficients of a block generally are scanned in a zig-zag fashion and transmitted using entropy coding methods. The 2×2 DC coefficients of the chroma component are scanned in raster-scan order. All inverse transform operations in H.264/AVC can be implemented using only additions and bit-shifting operations of 16-bit integer values. Similarly, only 16-bit memory accesses are needed for a good implementation of the forward transform and quantization process in the encoder.

[0212] The entropy encoder **800** in FIG. **13** in accordance with a coding arrangement described above with respect to FIG. **2**. A context modeler feeds a context model, i.e., a probability information, to an arithmetic encoder, which is also referred to as the regular coding engine. The to be encoded bit, i.e. a bin, is forwarded from the context modeler to the regular coding engine. This bin value is also fed back to the context modeler so that a context model update can be obtained. A bypass branch is provided, which includes an arithmetic encoder, which is also called the bypass coding engine. The bypass coding engine is operative to arithmetically encode the input bin values. Contrary to the regular coding engine, the bypass coding engine is not an adaptive coding engine but works preferably with a fixed probability model without any context adaption. A selection of the two branches can be obtained by means of switches. The binarizer device is operative to binarize non-binary valued syntax elements for obtaining a bin string, i.e., a string of binary values. In case the syntax element is already a binary value syntax element, the binarizer is bypassed.

[0213] Therefore, in CABAC (CABAC=Context-based Adaptive Binary Arithmetic Coding) the encoding process consists of at most three elementary steps:

[0214] 1. binarization

[0215] 2. context modeling

[0216] 3. binary arithmetic coding

[0217] In the first step, a given non-binary valued syntax element is uniquely mapped to a binary sequence, a so-called bin string. When a binary valued syntax element is given, this initial step is bypassed, as shown in FIG. **2**. For each element of the bin string or for each binary valued syntax element, one or two subsequent steps may follow depending on the coding mode.

[0218] In the co-called regular coding mode, prior to the actual arithmetic coding process the given binary decision, which, in the sequel, we will refer to as a bin, enters the context modeling stage, where a probability model is selected such that the corresponding choice may depend on previously encoded syntax elements or bins. Then, after the assignment of a context model the bin value along with its associated model is passed to the regular coding engine, where the final

stage of arithmetic encoding together with a subsequent model updating takes place (see FIG. **2**).

[0219] Alternatively, the bypass coding mode is chosen for selected bins in order to allow a speedup of the whole encoding (and decoding) process by means of a simplified coding engine without the usage of an explicitly assigned model. This mode is especially effective when coding the bins of the primary suffix of those syntax elements, concerning components of differences of motion vectors and transform coefficient levels.

[0220] In the following, the three main functional building blocks, which are binarization, context modeling, and binary arithmetic coding in the encoder of FIG. **13**, along with their interdependencies are discussed in more detail.

[0221] In the following, several details on binary arithmetic coding will be set forth.

[0222] Binary arithmetic coding is based on the principles of recursive interval subdivision that involves the following elementary multiplication operation. Suppose that an estimate of the probability $p_{LPS} \epsilon (0, 0.5]$ of the least probable symbol (LPS) is given and that the given interval is represented by its lower bound L and its width (range) R. Based on that settings, the given interval is subdivided into two subintervals: one interval of width

$$R_{LPS}=R \times p_{LPS},$$

which is associated with the LPS, and the dual interval of width $R_{MPS}=R-R_{LPS}$, which is assigned to the most probable symbol (MPS) having a probability estimate of $1-p_{LPS}$. Depending on the observed binary decision, either identified as the LPS or the MPS, the corresponding sub-interval is then chosen as the new current interval. A binary value pointing into that interval represents the sequence of binary decisions processed so far, whereas the range of the interval corresponds to the product of the probabilities of those binary symbols. Thus, to unambiguously identify that interval and hence the coded sequence of binary decisions, the Shannon lower bound on the entropy of the sequence is asymptotically approximated by using the minimum precision of bits specifying the lower bound of the final interval.

[0223] An important property of the arithmetic coding as described above is the possibility to utilize a clean interface between modeling and coding such that in the modeling stage, a model probability distribution is assigned to the given symbols, which then, in the subsequent coding stage, drives the actual coding engine to generate a sequence of bits as a coded representation of the symbols according to the model distribution. Since it is the model that determines the code and its efficiency in the first place, it is of importance to design an adequate model that explores the statistical dependencies to a large degree and that this model is kept "up to date" during encoding. However, there are significant model costs involved by adaptively estimating higher-order conditional probabilities.

[0224] Suppose a pre-defined set T of past symbols, a so-called context template, and a related set $C=\{0, \ldots, C-1\}$ of contexts is given, where the contexts are specified by a modeling function F. For each symbol x to be coded, a conditional probability $p(x|F(z))$ is estimated by switching between different probability models according to the already coded neighboring symbols $z \epsilon T$. After encoding x using the estimated conditional probability $p(x|F(z))$ is estimated on the fly by tracking the actual source statistics. Since the number of different conditional probabilities to be estimated for an alphabet size of m is high, it is intuitively clear that the model cost, which represents the cost of "learning" the model distribution, is proportional to the number of past symbols to the power of four.

[0225] This implies that by increasing the number C of different context models, there is a point, where overfitting of the model may occur such that inaccurate estimates of $p(x|F(z))$ will be the result.

[0226] This problem is solved in the encoder of FIG. **12** by imposing two severe restrictions on the choice of the context models. First, very limited context templates T consisting of a few neighbors of the current symbol to encode are employed such that only a small number of different context models C is effectively used.

[0227] Secondly, context modeling is restricted to selected bins of the binarized symbols and is of especially advantage with respect to primary prefix and suffix of the motion vector differences and the transform coefficient levels but which is also true for other syntax elements. As a result, the model cost is drastically reduced, even though the ad-hoc design of context models under these restrictions may not result in the optimal choice with respect to coding efficiency.

[0228] Four basic design types of context models can be distinguished. The first type involves a context template with up to two neighboring syntax elements in the past of the current syntax element to encode, where the specific definition of the kind of neighborhood depends on the syntax element. Usually, the specification of this kind of context model for a specific bin is based on a modeling function of the related bin values for the neighboring element to the left and on top of the current syntax element, as shown in FIG. **13**, and as was described above with respect to FIG. **5-12**. This design type of context modeling corresponds to the above description.

[0229] The second type of context models is only defined for certain data subtypes. For this kind of context models, the values of prior coded bins $(b_0, b_1, b_2, \ldots, b_{i-1})$ are used for the choice of a model for a given bin with index i. Note that these context models are used to select different models for different internal nodes of a corresponding binary tree.

[0230] Both the third and fourth type of context models is applied to residual data only. In contrast to all other types of context models, both types depend on context categories of different block types. Moreover, the third type does not rely on past coded data, but on the position in the scanning path. For the fourth type, modeling functions are specified that involve the evaluation of the accumulated number of encoded (decoded) levels with a specific value prior to the current level bin to encode (decode).

[0231] Besides these context models based on conditional probabilities, there are fixed assignments of probability models to bin indices for all those bins that have to be encoded in regular mode and to which no context model of the previous specified category can be applied.

[0232] The above described context modeling is suitable for a video compression engine such as video compression/decompression engines designed in accordance with the presently emerging H.264/AVC video compression standard. To summarize, for each bin of a bin string the context modeling, i.e., the assignment of a context variable, generally depends on the to be processed data type or sub-data type, the precision of the binary decision inside the bin string as well as the values of previously coded syntax elements or bins. With the exception of special context variables, the probability model

of a context variable is updated after each usage so that the probability model adapts to the actual symbol statistics.

[0233] A specific example for a context-based adaptive binary arithmetic coding scheme to which the assignment of context model of the above embodiments could be applied is described in: D. Marpe, G. Blättermann, and T. Wiegand, "Adaptive codes for H.26L," ITU-T SG16/Q.6 Doc. VCEG-L13, Eibsee, Germany, Jan. 2003-07-10.

[0234] It is noted that the above described steps in the above described flow charts could be implemented in software, for example in individual routines, or in Hardware, for example in an ASIC.

[0235] While this invention has been described in terms of several preferred embodiments, there are alterations, permutations, and equivalents which fall within the scope of this invention. It should also be noted that there are many alternative ways of implementing the methods and compositions of the present invention. It is therefore intended that the following appended claims be interpreted as including all such alterations, permutations, and equivalents as fall within the true spirit and scope of the present invention.

What is claimed is:

1. A method for encoding a video signal representing at least one video frame, with at least one video frame being composed of picture samples, the picture samples belonging either to a first or a second field being captured at different time instants, the video frame being spatially divided up into macroblock pair regions, each macroblock pair region being associated with a top and bottom macroblock, the method comprising the following steps:

deciding, for each macroblock pair region, as to whether same is of a first or a second distribution type;

assigning, for each macroblock pair region, each of the pixel samples in the respective macroblock pair region to a respective one of the top and bottom macroblock of the respective macroblock pair region, in accordance with the distribution type of the respective macroblock pair region;

pre-coding the video signal into a pre-coded video signal, the pre-coding comprising the sub-step of pre-coding a current macroblock of the top and bottom macroblock associated with a current macroblock pair region of the macroblock pair regions to obtain a current syntax element being pattern indicator specifying which of sub-parts of the current macroblock may contain non-zero transform coefficients and which may not;

determining, for the current syntax element, a neighboring macroblock at least based upon as to whether the current macroblock pair region is of a first or second distribution type;

assigning one of at least two context models to the current syntax element based on a pre-determined attribute of the neighboring macroblock, wherein each context model is associated with a different probability estimation; and

arithmetically encoding the syntax element into a coded bit stream based on the probability estimation with which the assigned context model is associated.

2. The method in accordance with claim **1**, wherein the step of determining is further dependent on as to whether the current macroblock is the top or the bottom macroblock of the current macroblock pair region.

3. The method in accordance with claim **1**, wherein the step of pre-coding further comprises the step of pre-coding the reference sub-part to obtain a reference syntax element as the predetermined attribute.

4. The method in accordance with claim **1**, wherein the step of determining is performed twice in order to obtain a neighboring macroblock to the top and a neighboring macroblock to the left of the current macroblock and the step of assigning one of at least two context models is based on a pre-determined attribute of both the neighboring macroblock to the top and the neighboring macroblock to the left of the current macroblock.

5. A method for decoding a predetermined syntax element from a coded bit stream, the coded bit stream being an arithmetically encoded version of a pre-coded video signal, the pre-coded video signal being a pre-coded version of a video signal, the video signal representing at least one video frame being composed of picture samples, the picture samples belonging either to a first or a second field being captured at a different time instants, the video frame being spatially divided up into macroblock pair regions, each macroblock pair region being associated with a top and a bottom macroblock, each macroblock pair region being either of a first or a second distribution type, wherein, for each macroblock pair region, each of the pixel samples in the respective macroblock pair region is assigned to a respective one of the top and bottom macroblock of the respective macroblock pair region in accordance with the distribution type of the respective macroblock pair region, wherein the predetermined syntax element relates to a predetermined macroblock of the top and bottom macroblock of a predetermined macroblock pair region of the macroblock pair regions and is a pattern indicator specifying which of sub-parts of the predetermined macroblock may contain non-zero transform coefficients and which may not, wherein the method comprises the following steps:

determining, for the predetermined syntax element, a neighboring macroblock at least based upon as to whether the predetermined macroblock pair region is of a first or a second distribution type;

assigning one of at least two context models to the predetermined syntax element based on a predetermined attribute of the neighboring macroblock, wherein each context model is associated with a different probability estimation; and

arithmetically decoding the predetermined syntax element from the coded bit stream based on the probability estimation with which the assigned context model is associated.

6. The method in accordance with claim **5**, wherein the step of determining is performed among a set comprising the other macroblock of the top and bottom macroblock of the predetermined macroblock pair region, the top macroblock of a macroblock pair region adjacent to the predetermined macroblock pair region and the bottom macroblock of the macroblock pair region adjacent to the predetermined macroblock pair region.

7. The method in accordance with claim **5**, wherein the step of determining is performed among a set comprising at least one of a top and a bottom macroblock of a macroblock pair region adjacent to the predetermined macroblock pair region, and is further dependent on as to whether the macroblock pair region adjacent to the predetermined macroblock pair region is of the first or the second distribution type.

8. Method in accordance with claim **5**, wherein the assignment of the pixel samples is such that, in accordance with the first distribution type, pixel samples lying in an upper part of the respective macroblock pair region are assigned to the top macroblock and pixel samples lying in a lower part of the respective macroblock pair region are assigned to the bottom macroblock, whereas, in accordance with the second distribution type, pixel samples in the respective macroblock pair region belonging to the first field are assigned to the top macroblock and pixel samples in the respective macroblock pair region belonging to the second field are assigned to the bottom macroblock.

9. The methods in accordance with claim **8**, wherein the pixel samples belonging to the first field and the pixel samples belonging to the second field are interlaced row-wise so that pixel samples assigned to the top and bottom macroblock of a respective macroblock pair region in accordance with the first distribution type have a greater pitch in row direction than pixel samples assigned to the top and bottom macroblock of a respective macroblock pair region in accordance with the second distribution type.

10. The method in accordance with claim **5**, wherein the step of determining is further dependent on as to whether the predetermined macroblock is the top or the bottom macroblock of the predetermined macroblock pair region.

11. Method in accordance with claim **5**, wherein the step of arithmetically decoding comprises:

binary arithmetically decoding a sequence of the binary decisions belonging to the predetermined syntax element, wherein, at least for a predetermined part of the binary decisions, the decoding is based on the probability estimation with which the assigned context model is associated; and

reconstructing the syntax element from the sequence of binary decisions.

12. The method in accordance with claim **1**, wherein the step of determining is performed twice in order to obtain a neighboring macroblock to the top and a neighboring macroblock to the left of the predetermined macroblock and the step of assigning one of at least two context models is based on a pre-determined attribute of both the neighboring macroblock to the top and the neighboring macroblock to the left of the predetermined macroblock.

13. An Apparatus for encoding a video signal representing at least one video frame, with at least one video frame being composed of picture samples, the picture samples belonging either to a first or a second field being captured at different time instants, the video frame being spatially divided up into macroblock pair regions, each macroblock pair region being associated with a top and bottom macroblock, the apparatus comprising

means for deciding, for each macroblock pair region, as to whether same is of a first or a second distribution type;

means for assigning, for each macroblock pair region, each of the pixel samples in the respective macroblock pair region to a respective one of the top and bottom macroblock of the respective macroblock pair region, in accordance with the distribution type of the respective macroblock pair region;

means for pre-coding the video signal into a pre-coded video signal, the pre-coding comprising the sub-step of pre-coding a current macroblock of the top and bottom macroblock associated with a current macroblock pair region of the macroblock pair regions to obtain a current syntax element being a pattern indicator specifying which of sub-parts of the predetermined macroblock may contain non-zero transform coefficients and which may not;

means for determining, for the current syntax element, a neighboring macroblock at least based upon as to whether the current macroblock pair region is of a first or second distribution type;

means for assigning one of at least two context models to the current syntax element based on a pre-determined attribute of the neighboring macroblock, wherein each context model is associated with a different probability estimation; and

means for arithmetically encoding the syntax element into a coded bit stream based on the probability estimation with which the assigned context model is associated.

14. An apparatus for decoding a predetermined syntax element from a coded bit stream, the coded bit stream being an arithmetically encoded version of a pre-coded video signal, the pre-coded video signal being a pre-coded version of a video signal, the video signal representing at least one video frame being composed of picture samples, the picture samples belonging either to a first or a second field being captured at a different time instants, the video frame being spatially divided up into macroblock pair regions, each macroblock pair region being associated with a top and a bottom macroblock, each macroblock pair region being either of a first or a second distribution type, wherein, for each macroblock pair region, each of the pixel samples in the respective macroblock pair region is assigned to a respective one of the top and bottom macroblock of the respective macroblock pair region in accordance with the distribution type of the respective macroblock pair region, wherein the predetermined syntax element relates to a predetermined macroblock of the top and bottom macroblock of a predetermined macroblock pair region of the macroblock pair regions and is a pattern indicator specifying which of sub-parts of the predetermined macroblock may contain non-zero transform coefficients and which may not, wherein the apparatus comprises

means for determining, for the predetermined syntax element, a neighboring macroblock at least based upon as to whether the predetermined macroblock pair region is of a first or a second distribution type;

means for assigning one of at least two context models to the predetermined syntax element based on a predetermined attribute of the neighboring macroblock, wherein each context model is associated with a different probability estimation; and

means for arithmetically decoding the predetermined syntax element from the coded bit stream based on the probability estimation with which the assigned context model is associated.

15. Method according to claim **5**, wherein the step of assigning one of at least two context models is performed such that the assignment is based on

an availability of the neighboring macroblock indicating as to whether the predetermined macroblock and the neighboring macroblock belong to the same slice of the video frame or to different slices of the video frame.

16. Method according to claim **5**, wherein the step of assigning one of at least two context models is performed such that the assignment is based on

a macroblock type indicator of the neighboring macroblock, specifying a partitioning of the neighboring macroblock used for motion prediction.

17. Method according to claim **5**, wherein the step of assigning one of at least two context models is performed such that the assignment is based on

the neighboring macroblock being inter or intra coded in the coded bit stream.

18. Method according to claim **5**, wherein the step of assigning one of at least two context models is performed such that the assignment is based on

a syntax element specifying, for the neighboring partition, which of sub-parts of the neighboring macroblock may contain non-zero transform coefficients and which may not.

19. Method according to claim **5**, wherein the step of assigning one of at least two context models is performed such that the assignment is based on

an availability of the neighboring macroblock indicating as to whether the predetermined macroblock and the neighboring macroblock belong to the same slice of the video frame or to different slices of the video frame,

a macroblock type indicator of the neighboring macroblock, specifying a partitioning of the neighboring macroblock used for motion prediction,

the neighboring macroblock being inter or intra coded in the coded bit stream, and

a syntax element specifying, for the neighboring partition, which of sub-parts of the neighboring macroblock may contain non-zero transform coefficients and which may not.

20. Method according to claim **9**, wherein the step of determining is performed such that the neighboring macroblock belongs to a macroblock pair region to the left of the predetermined macroblock pair region of predetermined macroblock and the determination results in a selection among the top and bottom macroblock of macroblock pair region to the left of the predetermined macroblock pair region that is effectively dependent on

the predetermined macroblock pair region being of the first or second distribution type,

the macroblock pair region to the left being of the first or second distribution type, and

the predetermined macroblock being the top or bottom macroblock of the predetermined macroblock pair region.

* * * * *