(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 April 2020 (30.04.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/083108 A1

(51) International Patent Classification:
H04N 19/70 (2014.01)

(21) International Application Number:
PCT/CN2019/111895

(22) International Filing Date:
18 October 2019 (18.10.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

62/749,145	23 October 2018 (23.10.2018)	US
62/750,282	25 October 2018 (25.10.2018)	US
62/768,200	16 November 2018 (16.11.2018)	US

(71) Applicant: **MEDIATEK INC.** [CN/CN]; No. 1, Dusing 1st Rd., Hsinchu Science Park, Hsinchu City, Taiwan 30078 (CN).

(72) Inventors: **CHEN, Ching-Yeh**; No. 1, Dusing 1st Rd., Hsinchu Science Park, Hsinchu City, Taiwan 30078 (CN). **CHUANG, Tzu-Der**; No. 1, Dusing 1st Rd., Hsinchu Science Park, Hsinchu City, Taiwan 30078 (CN). **SU, Yu-Chi**; No. 1, Dusing 1st Rd., Hsinchu Science Park, Hsinchu City, Taiwan 30078 (CN). **HSU, Chih-Wei**; No. 1, Dusing 1st Rd., Hsinchu Science Park, Hsinchu City, Taiwan 30078 (CN). **HUANG, Yu-Wen**; No. 1, Dusing 1st Rd., Hsinchu Science Park, Hsinchu City, Taiwan 30078 (CN).

(74) Agent: **BEIJING SANYOU INTELLECTUAL PROPERTY AGENCY LTD.**; 16th Fl., Block A, Corporate Square, No.35 Jinrong Street, Beijing 100033 (CN).

(54) Title: METHOD AND APPARATUS FOR REDUCTION OF IN-LOOP FILTER BUFFER

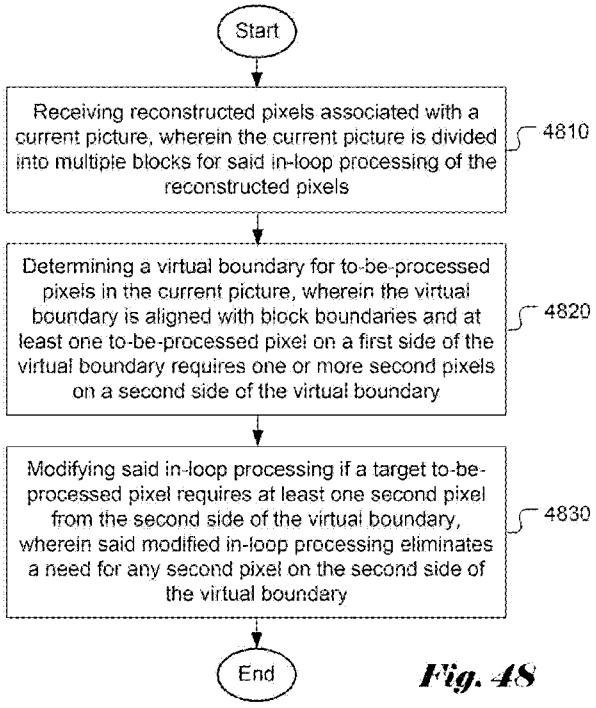

Fig. 48

(57) Abstract: Methods and apparatus for in-loop processing of reconstructed video are disclosed. According to one method, a virtual boundary is determined for to-be-processed pixels in the current picture, where the virtual boundary is aligned with block boundaries and at least one to-be-processed pixel on a first side of the virtual boundary requires one or more second pixels on a second side of the virtual boundary. According to the method, the in-loop processing is modified if a target to-be-processed pixel requires at least one second pixel from the second side of the virtual boundary and the modified in-loop processing eliminates the need for any second pixel on the second side of the virtual boundary. According to another method, the operations of block classification are changed when part of the required pixels in one 10x10 window used in classification are at the other side of virtual boundaries.

WO 2020/083108 A1

[Continued on next page]

(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

METHOD AND APPARATUS FOR REDUCTION OF IN-LOOP FILTER BUFFER

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] The present invention claims priority to U.S. Provisional Patent Application, Serial No. 62/749,145, filed October 23, 2018, 62/750,282, filed October 25, 2018 and 62/768,200, filed November 16, 2018. The U.S. Provisional Patent Applications are hereby incorporated by reference in their entireties.

FIELD OF THE INVENTION

[0002] The present invention relates to filter processing in a video coding system. In particular, the present invention relates to method and apparatus are related to in-loop filter processing across a tile or slice boundary in a video encoder or decoder.

BACKGROUND

[0003] Motion estimation is an effective inter-frame coding technique to exploit temporal redundancy in video sequences. Motion-compensated inter-frame coding has been widely used in various international video coding standards The motion estimation adopted in various coding standards is often a block-based technique, where motion information such as coding mode and motion vector is determined for each macroblock or similar block configuration. In addition, intra-coding is also adaptively applied, where the picture is processed without reference to any other picture. The inter-predicted or intra-predicted residues are usually further processed by transformation, quantization, and entropy coding to generate a compressed video bitstream. During the encoding process, coding artefacts are introduced, particularly in the quantization process. In order to alleviate the coding artefacts, additional processing has been applied to reconstructed video to enhance picture quality in newer coding systems. The additional processing is often configured in an in-loop operation so that the encoder and decoder may derive the same reference pictures to achieve improved system performance.

[0004] Fig. 1A illustrates an exemplary adaptive Inter/Intra video coding system incorporating in-loop processing. For inter-prediction, Motion Estimation (ME)/Motion Compensation (MC) 112 is used to provide prediction data based on video data from other picture or pictures. Switch 114 selects Intra Prediction 110 or inter-prediction data and the selected prediction data is supplied to Adder 116 to form prediction errors, also called residues. The prediction error is then processed by Transformation (T) 118 followed by Quantization (Q) 120. The transformed and quantized residues are then coded by Entropy Encoder 122 to form a video bitstream corresponding to the compressed video data. The bitstream associated with the transform coefficients is then packed with side information such as motion, mode, and other information associated with the image area. The side information may also be subject to entropy coding to reduce required bandwidth. Accordingly, the data associated with the side information are provided to Entropy Encoder 122 as shown in Fig. 1A. When an inter-prediction mode is used, a reference picture or pictures have to be reconstructed at the encoder end as well. Consequently, the transformed and quantized residues are processed by Inverse Quantization (IQ) 124 and Inverse Transformation (IT) 126 to recover the residues. The residues are then added back to prediction data 136 at Reconstruction (REC) 128 to reconstruct video data. The reconstructed video data may be stored in Reference Picture Buffer 134 and used for prediction of other frames.

[0005] As shown in Fig. 1A, incoming video data undergoes a series of processing in the encoding system.

The reconstructed video data from REC 128 may be subject to various impairments due to a series of processing. Accordingly, various in-loop processing is applied to the reconstructed video data before the reconstructed video data are stored in the Reference Picture Buffer 134 in order to improve video quality. In the High Efficiency Video Coding (HEVC) standard being developed, Deblocking Filter (DF) 130, Sample Adaptive Offset (SAO) 131 and Adaptive Loop Filter (ALF) 132 have been developed to enhance picture quality. The in-loop filter information may have to be incorporated in the bitstream so that a decoder can properly recover the required information. Therefore, in-loop filter information from SAO and ALF is provided to Entropy Encoder 122 for incorporation into the bitstream. In Fig. 1A, DF 130 is applied to the reconstructed video first; SAO 131 is then applied to DF-processed video; and ALF 132 is applied to SAO-processed video. However, the processing order among DF, SAO and ALF can be re-arranged.

[0006] A corresponding decoder for the encoder of Fig. 1A is shown in Fig. 1B. The video bitstream is decoded by Video Decoder 142 to recover the transformed and quantized residues, SAO/ALF information and other system information. At the decoder side, only Motion Compensation (MC) 113 is performed instead of ME/MC. The decoding process is similar to the reconstruction loop at the encoder side. The recovered transformed and quantized residues, SAO/ALF information and other system information are used to reconstruct the video data. The reconstructed video is further processed by DF 130, SAO 131 and ALF 132 to produce the final enhanced decoded video.

[0007] The coding process in HEVC is applied according to Largest Coding Unit (LCU). The LCU is adaptively partitioned into coding units using quadtree. In each leaf CU, DF is performed for each 8x8 block and in HEVC Test Model Version 4.0 (HM-4.0), the DF is applies to 8x8 block boundaries. For each 8x8 block, horizontal filtering across vertical block boundaries is first applied, and then vertical filtering across horizontal block boundaries is applied. During processing of a luma block boundary, four pixels of each side are involved in filter parameter derivation, and up to three pixels on each side can be changed after filtering. For horizontal filtering across vertical block boundaries, unfiltered reconstructed pixels (i.e., pre-DF pixels) are used for filter parameter derivation and also used as source pixels for filtering. For vertical filtering across horizontal block boundaries, unfiltered reconstructed pixels (i.e., pre-DF pixels) are used for filter parameter derivation, and DF intermediate pixels (i.e. pixels after horizontal filtering) are used for filtering. For DF processing of a chroma block boundary, two pixels of each side are involved in filter parameter derivation, and at most one pixel on each side is changed after filtering. For horizontal filtering across vertical block boundaries, unfiltered reconstructed pixels are used for filter parameter derivation and are used as source pixels for filtering. For vertical filtering across horizontal block boundaries, DF processed intermediate pixels (i.e. pixels after horizontal filtering) are used for filter parameter derivation and also used as source pixel for filtering.

[0008] Sample Adaptive Offset (SAO) 131 is also adopted in HM-4.0, as shown in Fig. 1A. SAO can be regarded as a special case of filtering where the processing only applies to one pixel. In SAO, pixel classification is first done to classify pixels into different groups (also called categories or classes). The pixel classification for each pixel is based on a 3x3 window. Upon the classification of all pixels in a picture or a region, one offset is derived and transmitted for each group of pixels. ALF is not adopted by the HEVC (High Efficiency Video Coding). However, ALF is being considered for the emerging video coding standard, names VVC (Versatile Video Coding). The filter coefficients of ALF are derived by minimizing the sum of the distortion between filtered samples and original samples. Furthermore, the derived filter coefficients are signalled in the bitstream

with on/off control flags. Multiple filters can be used in one slice and the filter selection includes implicit selection by block-based classification and explicit selection by signalled syntax.

[0009] In HM-4.0, DF is applied on 8x8 block boundaries to reduce the visibility of artefacts at block boundaries. Fig. 2 shows a vertical boundary 212 in block 210 and a horizontal boundary 222 in block 220, where 8x8 blocks are partly shown (4x8 or 8x4). In one picture, all vertical boundaries can be horizontally filtered in parallel, and then all horizontal boundaries can be vertically filtered in parallel. During processing of a luma boundary, four pixels of each side (p_0-p_3, q_0-q_3) are involved in filter parameter derivation, and at most three pixels of each side (p_0-p_2, q_0-q_2) can be changed after filtering. For luma horizontal DF, pre-DF pixels (i.e. pixels before horizontal DF) are used for deriving filter parameters and filtering. For luma vertical DF, pre-DF pixels are used for deriving filter parameters, and H-DF pixels (i.e. pixels after horizontal DF) are used for filtering. During processing of a chroma boundary, two pixels of each side (p_0-p_1, q_0-q_1) are involved in filter parameter derivation, and at most one pixel of each side (p_0, q_0) is changed after filtering. For chroma horizontal DF, pre-DF pixels are used for deriving filter parameters and filtering. For chroma vertical DF, H-DF pixels are used for deriving filter parameters and filtering.

[0010] In HM-4.0, SAO is applied to luma and chroma components, and each of the luma components is independently processed. SAO can divide one picture into multiple LCU-aligned regions, and each region can select one SAO type among two Band Offset (BO) types, four Edge Offset (EO) types, and no processing (OFF). For each to-be-processed (also called to-be-filtered) pixel, BO uses the pixel intensity to classify the pixel into a band. The pixel intensity range is equally divided into 32 bands as shown in Fig. 3. After pixel classification, one offset is derived for all pixels of each band, and the offsets of center 16 bands or outer 16 bands are selected and coded. As for EO, it uses two neighboring pixels of a to-be-processed pixel to classify the pixel into a category. The four EO types correspond to 0°, 90°, 135°, and 45° as shown in Fig. 4. Similar to BO, one offset is derived for all pixels of each category except for category 0, where Category 0 is forced to use zero offset. Table 1 shows the EO pixel classification, where "C" denotes the pixel to be classified.

Table 1.

Category	Condition
1	C < two neighbors
2	C < one neighbor && C == one neighbor
3	C > one neighbor && C == one neighbor
4	C > two neighbors
0	None of the above

[0011] In HM-4.0, ALF has two filter shape options, cross11x5 (510) and snowflake5x5 (520), for luma and chroma, as shown in Fig. 5. In each picture, the luma component can choose one shape, and the chroma components can choose one shape. Up to 16 luma filters and at most one chroma filter can be applied for each picture. In order to allow localization of ALF, there are two modes for luma pixels to select filters. One is a region-based adaptation (RA) mode, and the other is a block-based adaptation (BA) mode. The RA mode divides one luma picture into 16 regions. Once the picture size is known, the 16 regions are determined and fixed. The

regions can be merged, and one filter is used for each region after merging. The BA mode uses edge activity and direction as a property for each 4x4 block. Calculating the property of a 4x4 block (610) requires 5x5 pixels (612), as shown in Fig. 6A. After the properties of 4x4 blocks are calculated, they are classified into 15 categories. The categories can be merged, and one filter is used for each category after merging. As for the chroma components, since they are relatively flat, no local adaptation is used, and the two components of a picture share one filter.

[0012] In the emerging VVC (Versatile Video Coding) standard being developed, more flexible ALF has been disclosed to improve the performance. For block-based classification, one picture is partitioned into several 4x4 luma blocks and one group index is derived for each 4x4 luma block. The group index is used to select luma filter from a filter set. In order to reduced required data for the filter coefficients, the filter coefficients may be rotated based on characteristics of one 4x4 block so that some coefficients don't need to be transmitted. In VVC, up to 25 groups can be used for ALF of one luma filter set, which is associated with 5 activity levels based on the gradients magnitude and 5 directions based on the gradients direction. Different groups can share one filter, where multiple groups can be merged into one merged group and one merged group has one filter.

[0013] In order to determine the Group index, the gradients for each 4x4 block are calculated. The gradients include four types corresponding to $Grad_H$: Horizontal gradient, $Grad_V$:Vertical gradient, $Grad_{45}$:Diagonal45 gradient, and $Grad_{135}$: Diagonal135 gradient. The gradient for horizontal, vertical, diagonal 45° and diagonal 135° use three consecutive pixels in respective directions with the center weight equal to -2 and two side weights equal to 1. Block classification is evaluated at selected locations of an evaluation block 620 as shown in Fig. 6B, where the target 4x4 block 622 shown in gray colour is located in the center of the evaluation block. The 8x8 evaluation block is formed by extended 2 pixels on each direction of the 4x4 block. Gradients at 32 positions (i.e., *selected locations*) are used to calculate the summation of gradients for one 4x4 block, where the selected locations are indicated by small boxes with gradient locations (e.g. $D_{i,j}$). A total of 32 gradients are accumulated for each type and accessing 10x10 samples is required to derive gradients since one additional pixel in each direction is needed for calculating the gradients. As shown in Fig. 6B, the selected locations correspond to 2:1 horizontal sub-sampled locations with one-pixel offset between even lines and odd lines

[0014] In order to determine the Group index, the following two ratios are further calculated:

$$\text{Ratio}_{HV} = \max(Grad_H, Grad_V) / \min(Grad_H, Grad_V),$$

$$\text{Ratio}_{Dia} = \max(Grad_{135}, Grad_{45}) / \min(Grad_{135}, Grad_{45})$$

[0015] The direction index (*Direction*) is determined according to:

if ($\text{Ratio}_{HV} < 2$ && $\text{Ratio}_{Dia} < 2$) *Direction* = 0

else if ($\text{Ratio}_{HV} > \text{Ratio}_{Dia}$) *Direction* = ($\text{Ratio}_{HV} > 4.5$) ? 2 : 1

else *Direction* = ($\text{Ratio}_{Dia} > 4.5$) ? 4 : 3

[0016] The 5 activity levels is determined by quantizing the sum of (GradH + GradV) into 5 levels (*Activity*). The Group index is calculated from (5 x *Direction* + *Activity*).

[0017] An example of 7x7 diamond filter for luma and 5x5 diamond filter for chroma component are shown

in Fig. 6C, where the coefficients are half symmetry across diagonal line 630. In Fig. 6B, coefficients 0-6 are for the 5x5 diamond filter and coefficients 0-12 are for the 7x7 diamond filter. The coefficients are represented in 8 bits with 1 bit for sign and 7 bits for the fractional part. Non-center coefficients are signalled and in the range of [-128, 127] with one clipping threshold, if needed. Center coefficient is set equal to 1.

[0018] The line buffer requirement for ALF in VTM 4.0 is shown in Fig. 6D. Filter footprint is 7x7 diamond filter for luma and 5x5 diamond filter for chroma. A total of 6 lines are required for luma filtering process. A total of 4 lines are required for chroma filter process.

[0019] The line buffer requirement 4x4 block-based classification for ALF in VTM 4.0 is shown in Fig. 6E. Classification is performed for 4x4 block. One 4x4 block needs a 10x10 window, which is included in 7x7 diamond filter. If unavailable regions for ALF is aligned with 4x4 grid, then only 7 lines are required.

[0020] In order to reduce the line buffer usage, virtual boundary is proposed during HEVC standard development. The concept is when ALF is applied to the pixels at one side of virtual boundary, then the pixels at the other side cannot be used as show in Fig. 6F.

[0021] When LCU-based processing is used for DF, SAO, and ALF, the entire decoding process can be done LCU by LCU in a raster scan with an LCU-pipelining fashion for parallel processing of multiple LCUs. In this case, line buffers are required for DF, SAO, and ALF because processing one LCU row requires pixels from the above LCU row. If off-chip line buffers (e.g. DRAM) are used, the external memory bandwidth and power consumption will be increased; if on-chip line buffers (e.g. SRAM) are used, the chip area will be increased. Therefore, although line buffers are already much smaller than picture buffers, it is still desirable to reduce line buffers.

[0022] Fig. 7 explains the number of luma line buffers required for DF, SAO, and ALF with LCU-based decoding. Given a current LCU (lower H. LCU boundary 710 and left V. LCU boundary 712), lines A-J are first processed by horizontal DF and then by vertical DF. Next, it is suggested to delay the horizontal DF for lines K-N until the lower LCU is available in order to avoid line buffers of H-DF pixels for the vertical DF for lines K-N, which cannot be processed at this moment because the lower LCU is not yet available and DF needs four pre-DF pixels and four H-DF pixels on each side of the horizontal boundary for deriving filter parameters and filtering, respectively, as illustrated by the 4-pixel stripe (720). Therefore, four lines (K-N) of pre-DF pixels have to be stored for DF. Next, SAO is going to be applied on DF output pixels. Since the vertical DF for lines K-N will not change line K, horizontal DF can be additionally applied on line K for SAO to process line J, as illustrated by the 3x3 square (730). Please note that the H-DF pixels of line K will not be stored in the line buffer and have to be generated again when the lower LCU comes, which is not a problem in hardware. After SAO processes lines A-J, the 4x4 block property, as illustrated by the 4x4 square (740), can be calculated. The 5x5 supporting area (742) is indicated. At this moment ALF can process lines A-H. After this, no further process can be done for the current LCU until the lower LCU comes. When the lower LCU comes, lines K-P are first processed by DF, and then lines K-P are processed by SAO. When SAO is applied to line K, line J is required. Therefore, one line (J) of DF output pixels has to be stored for SAO. Next, the 4x4 block properties for lines I-P are calculated. Finally, lines I-P can be filtered by ALF. When line I is filtered, it requires lines G-K, as illustrated by the filter shape (750) and the center of the filter (752) is indicated. However, calculating block properties of lines I-J still needs lines F-J. Therefore, five lines (F-J) of SAO output pixels have to be stored for ALF. In total, the entire in-loop filtering requires 10 luma line buffers. When the entire decoding system is

considered, since the Intra luma prediction already stores one line (N) of pre-DF pixels, this luma line buffer can be shared. Moreover, if a filter index line buffer can be used to store BA mode filter selections for lines G-J, it is not necessary to compute the block properties again during filtering of lines I-J. In this way, one line (F) of SAO output pixels can be saved for ALF. The filter index line buffer requires only 4 bits per 4x4 block, which is only about 10% in comparison with the pixel line buffer.

[0023] Fig. 8 explains the number of chroma line buffers required for DF, SAO, and ALF with LCU-based decoding. Given a current LCU (H. LCU boundary 810 and V. LCU boundary 812), lines A-M are first processed by horizontal DF, and then lines A-L are processed by vertical DF. Next, it is suggested to delay the horizontal DF for lines N until the lower LCU comes in order to share the line buffer of pre-DF pixels of line N with intra chroma prediction. As for the vertical DF for lines M-N, it has to wait for the lower LCU. Please recall that chroma DF reads two H-DF pixels and may write one pixel on each side of the boundary, as illustrated by the 2-pixel stripe. Therefore, one line (M) of H-DF pixels and one line (N) of pre-DF pixels have to be stored for DF. Next, SAO is going to be applied on DF output pixels. Since the vertical DF for lines M-N will not change line M, H-DF pixels of line M are also DF output pixels of line M, and SAO can process line L, as illustrated by the 3x3 square (820). After SAO processes lines A-L, ALF can process lines A-J. After this, no further process can be done for the current LCU until the lower LCU comes. When the lower LCU comes, lines N-P are first processed by horizontal DF, and then lines M-P are processed by vertical DF and by SAO. When SAO processes line M, line L is required. Therefore, one line (L) of DF output pixels has to be stored for SAO. Finally, lines K-P can be filtered by ALF. When line K is filtered, it requires lines I-M, as illustrated by the filter shape (820) with the center of the filter (822) is indicated. Therefore, four lines (I-L) of SAO output pixels have to be stored for ALF. In total, the entire in-loop filtering requires seven chroma line buffers.

[0024] Virtual Boundaries to Reduce Line Buffer Usage

[0025] In order to eliminate the line buffer requirements of SAO and ALF, we introduce the concept of virtual boundary (VB). As shown in Fig. 9A, VBs are upward shifted horizontal LCU boundaries by N pixels. In Fig. 9B, VBs are left shifted vertical LCU boundaries by N pixels. For each LCU, SAO and ALF can process pixels above the VB before the lower LCU comes but cannot process pixels below the VB until the lower LCU comes, which is caused by DF. With consideration of the DF in HM-4.0, the space between the proposed VB and the horizontal LCU boundary is set as three pixels for luma (i.e. N=3 in Fig. 9A and Fig. 9B) and one pixel for chroma (i.e. N=1 in Fig. 9A and Fig. 9B). SAO and ALF are modified to ensure that every to-be-processed pixel on one side of a VB does not require any data access from the other side of the VB unless the data can become available in time without using any additional line buffer.

BRIEF SUMMARY OF THE INVENTION

[0026] A method and apparatus for in-loop processing of reconstructed video are disclosed. According to this method, reconstructed pixels associated with a current picture are received, where the current picture is divided into multiple blocks for in-loop processing of the reconstructed pixels. A virtual boundary is determined for to-be-processed pixels in the current picture, where the virtual boundary is aligned with block boundaries and at least one to-be-processed pixel on a first side of the virtual boundary requires one or more second pixels on a second side of the virtual boundary. According to the method, the in-loop processing is modified if a target to-be-processed pixel requires at least one second pixel from the second side of the virtual boundary and the

modified in-loop processing eliminates the need for any second pixel on the second side of the virtual boundary.

[0027] In one embodiment, the multiple blocks correspond to 4x4 blocks. In one embodiment, the multiple blocks correspond to smallest units in which all samples share the same in-loop processing. In one embodiment, the in-loop processing corresponds to SAO (Sample Adaptive Offset) processing or ALF (Adaptive Loop Filter) processing. In one embodiment, the reconstructed pixels correspond to luma pixels. For chroma reconstructed pixels of the current picture, a chroma virtual boundary is allowed to cross chroma blocks.

[0028] In one embodiment, if the target to-be-processed pixel requires any second pixel from the second side of the virtual boundary, such second pixel from the second side of the virtual boundary are padded. In another embodiment, if the target to-be-processed pixel requires any second pixel from the second side of the virtual boundary, such second pixel from the second side of the virtual boundary are padded using symmetric padding.

[0029] According to another embodiment, modified in-loop processing uses a modified filter with reduced footprint to eliminate the need for any second pixel on the second side of the virtual boundary.

[0030] The modified in-loop processing can be applied to the reconstructed pixels at an encoder side as well as a decoder side.

[0031] According to another method, an extended block covering the target block is determined for a target block belonging to the to-be-processed pixels. Block classification is derived for the target block based on the extended block, where if any target pixel of the extended block required for deriving the block classification is on the second side of the virtual boundary, at least one operation associated with the block classification is modified. The adaptive-loop-filter processing is then applied to the target block using the block classification with at least one operation modified if any target pixel of the extended block required for said deriving the block classification is on the second side of the virtual boundary.

[0032] In one embodiment, an evaluation block is determined for the target block, wherein the target block is located at a center of the evaluation block and surrounding pixels are added to the evaluation block to form the extended block. For example, the target block corresponds to a 4x4 block, the evaluation block corresponds to an 8x8 block and the extended block corresponds to a 10x10 block. In another embodiment, the target block corresponds to the smallest unit in which all samples share an identical adaptive-loop-filter processing.

[0033] In one embodiment, the block classification is derived based on gradients at selected locations of the evaluation block. In one embodiment, if a target gradient for a target selected location of the evaluation block requires any target pixel of the extended block on the second side of the virtual boundary, the target gradient is excluded from a summation of gradients for the block classification or is set to zero. In this case, the summation of gradients are normalized by a total number of available gradients at the selected locations of the evaluation block. Alternatively, the summation of gradients are weighted according to a total number of available gradients at the selected locations of the evaluation block.

[0034] In one embodiment, if a target gradient for a target selected location of the evaluation block requires any target pixel of the extended block on the second side of the virtual boundary, the target gradient for the target selected location of the evaluation block is modified to exclude said any target pixel of the extended block on the second side of the virtual boundary.

[0035] In one embodiment, the selected locations of the evaluation block correspond to 2:1 horizontal sub-sampled locations with one-pixel offset between even lines and odd lines.

BRIEF DESCRIPTION O.F THE DRAWINGS

[0036] Fig. 1A illustrates an exemplary adaptive inter/intra video encoding system incorporating DF, SAO and ALF in-loop processing.

[0037] Fig. 1B illustrates an exemplary adaptive inter/intra video decoding system incorporating DF, SAO and ALF in-loop processing.

[0038] Fig. 2 illustrates an example of deblocking filter process applied on 8x8 block boundaries to reduce the visibility of artefacts at block boundaries, where a vertical boundary and a horizontal boundary are shown.

[0039] Fig. 3 illustrates an example of band offset (BO), where the pixel intensity range is equally divided into 32 bands and an offset value is determined for each band.

[0040] Fig. 4 illustrates Edge Offset (EO) windows corresponding to 0°, 90°, 135°, and 45° used in HEVC (high efficiency video coding) standard to determine the category for a current pixel to apply SAO (Sample Adaptive Offset).

[0041] Fig. 5 illustrates an example of adaptive loop filter (ALF), where the ALF has two filter shape options corresponding to cross11x5 and snowflake5x5, for luma and chroma.

[0042] Fig. 6A illustrates an example of block adaptation (BA) for adaptive loop filter (ALF), where the BA mode uses edge activity and direction as a property for each 4x4 block.

[0043] Fig. 6B illustrates an example of block classification for a target block by evaluating gradients at selected locations of an evaluation block, where the target 4x4 block shown in gray colour is located in the center of the evaluation block.

[0044] Fig. 6C illustrates an example of 7x7 diamond filter for luma and 5x5 diamond filter for chroma component, where the coefficients are half symmetry across diagonal line.

[0045] Fig. 6D illustrates an example of the line buffer requirement for ALF in VTM 4.0, where the filter footprint is 7x7 diamond filter for luma and 5x5 diamond filter for chroma.

[0046] Fig. 6E illustrates an example of the line buffer requirement 4x4 block-based classification for ALF in VTM 4.0, where classification is performed for 4x4 block.

[0047] Fig. 6F illustrates an example of the concept is when ALF is applied to the pixels at one side of virtual boundary, then the pixels at the other side cannot be used.

[0048] Fig. 7 illustrates an example of the number of luma line buffers required for DF, SAO, and ALF with LCU-based decoding.

[0049] Fig. 8 illustrates an example of the number of chroma line buffers required for DF, SAO, and ALF with LCU-based decoding.

[0050] Fig. 9A illustrates an example of VBs by upward shifting horizontal LCU boundaries by N pixels.

[0051] Fig. 9B illustrates an example of VBs by left shifting vertical LCU boundaries by N pixels.

[0052] Fig. 10 illustrates an example VB processing for pixels above the VB, where all pixels of the current LCU have been processed by REC, and four lines (p_0-p_3) of pre-DF pixels are stored in DF line buffers.

[0053] Fig. 11 illustrates an example VB processing for pixels above the VB, where pixels above the VB are processed by horizontal DF.

[0054] Fig. 12 illustrates an example VB processing for pixels above the VB, where pixels above the VB are processed by vertical DF.

[0055] Fig. 13 illustrates an example VB processing for pixels above the VB, where pixels above the VB are going to be processed by SAO.

[0056] Fig. 14 illustrates an example VB processing for pixels above the VB, where all pixels above the VB have been processed by SAO.

[0057] Fig. 15 illustrates an example VB processing for pixels above the VB, where the block properties of lines p_4-p_7 are directly reused for line p_3 since SAO output pixels of lines p_0-p_2 are not yet available.

[0058] Fig. 16 illustrates an example VB processing for pixels above the VB, where snowflake5x5 is selected for ALF, SAO output pixels of the last row (line p_2) of the snowflake5x5 for filtering line p_4 and the last two rows (lines p_1-p_2) of the snowflake5x5 for filtering line p_3 are not yet available and are replaced by pre-DF pixels.

[0059] Fig. 17 illustrates an example VB processing for pixels above the VB, where cross9x9 is selected for ALF, SAO output pixels of the last row (line p_2) of the cross9x9 for filtering line p_6 and the last two rows (lines p_1-p_2) of the cross9x9 for filtering line p_5 are not yet available and are replaced by pre-DF pixels.

[0060] Fig. 18 illustrates an example VB processing for pixels above the VB, where cross9x9 is also selected for ALF, SAO output pixels of the last three rows (lines p_0-p_2) of the cross9x9 for filtering line p_4 are not yet available and are replaced by pre-DF pixels.

[0061] Fig. 19 illustrates an example VB processing for pixels above the VB, where all pixels above the VB have been processed by ALF. At this moment, pixels above the VB can be written to a decoded picture buffer before the lower LCU comes.

[0062] Figs. 20-29 illustrate the proposed luma VB processing for pixels below the VB.

[0063] Fig. 20 illustrates an example VB processing for pixels below the VB, where four lines (i.e., p_0-p_3) of pre-DF pixels can be read from the DF line buffers.

[0064] Fig. 21 illustrates an example VB processing for pixels below the VB, where pixels of line p_3 and pixels below the VB are processed by horizontal DF.

[0065] Fig. 22 illustrates an example VB processing for pixels below the VB, where the pixels below the VB are processed by vertical DF.

[0066] Fig. 23 illustrates an example VB processing for pixels below the VB, where the pixels below the VB are going to be processed by SAO.

[0067] Fig. 24 illustrates an example VB processing for pixels below the VB, where all pixels below the VB (and above the next VB) have been processed by SAO.

[0068] Fig. 25 illustrates an example VB processing for pixels below the VB, where SAO output pixels of line p_3 are replaced by the available DF output pixels.

[0069] Fig. 26 illustrates an example VB processing for pixels below the VB, where for filtering line p_2, SAO output pixels of the second row (i.e., line p_3) of the snowflake5x5 are replaced by DF output pixels; and SAO output pixels of the first row (i.e., line p_4) of the snowflake5x5 are replaced by nearest DF output pixels of line p_3.

[0070] Fig. 27 illustrates an example VB processing for pixels below the VB, where cross9x9 is selected for ALF; for filtering line p_2, the filter shape is reduced to cross9x3 with adding the six discarded coefficients to the center pixel; and SAO output pixels of the first row (i.e., line p_3) of the cross9x3 are replaced by DF output pixels.

[0071] Fig. 28 illustrates an example VB processing for pixels below the VB, where cross9x9 is also selected for ALF; for filtering line p_0, the filter shape is reduced to cross9x7 with adding the two discarded coefficients to the centre pixel; and SAO output pixels of the first row (i.e., line p_3) of the cross9x7 are replaced by DF output pixels.

[0072] Fig. 29 illustrates an example VB processing for pixels below the VB, where all pixels below the VB (and above the next VB) have been processed by ALF.

[0073] Figs. 30-38 show the chroma VB processing for pixels above the VB.

[0074] Figs. 39-47 show the chroma VB processing for pixels below the VB.

[0075] Fig. 48 illustrates a flowchart of an exemplary in-loop processing of reconstructed video according to an embodiment of the present invention.

[0076] Fig. 49 illustrates a flowchart of exemplary adaptive-loop-filter processing of reconstructed video according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0077] The following description is of the best-contemplated mode of carrying out the invention. This description is made for the purpose of illustrating the general principles of the invention and should not be taken in a limiting sense. The scope of the invention is best determined by reference to the appended claims.

[0078] The luma VB processing can be divided into two parts. The first part handles pixels above the VB, while the second part handles pixels below the VB. Figs. 10-19 illustrate various proposed luma VB processing for pixels above the VB. In Fig. 10, all pixels of the current LCU have been processed by REC, and four lines (p_0-p_3) of pre-DF pixels are stored in DF line buffers. In Fig. 11, pixels above the VB are processed by horizontal DF. Please recall that luma DF reads four pixels and writes up to three pixels on each side of the 8x8 block boundary. In Fig. 12, pixels above the VB are processed by vertical DF. In Fig. 13, pixels above the VB are going to be processed by SAO. Originally, SAO processing of line p_3 may need DF output pixels of line p_2. Since DF output pixels of line p_2 are not yet available, they are replaced by pre-DF pixels of line p_2 for SAO processing of line p_3. In Fig. 14, all pixels above the VB have been processed by SAO. In Figs. 15-18, pixels above the VB are going to be processed by ALF. Originally, calculating ALF block properties of line p_3 requires SAO output pixels of lines p_0-p_4. Since SAO output pixels of lines p_0-p_2 are not yet available, the block properties of lines p_4-p_7 are directly reused for line p_3, as shown in Fig. 15. During filtering, SAO output pixels below the VB may also be originally needed. In these cases, SAO output pixels are replaced by pre-DF pixels. In Fig. 16, where snowflake5x5 is selected for ALF, SAO output pixels of the last row (i.e., line p_2) of the snowflake5x5 for filtering line p_4 and the last two rows (i.e., lines p_1-p_2) of the snowflake5x5 for filtering line p_3 are not yet available and are replaced by pre-DF pixels. In Fig. 17, where cross9x9 is selected for ALF, SAO output pixels of the last row (i.e., line p_2) of the cross9x9 for filtering line p_6 and the last two rows (i.e., lines p_1-p_2) of the cross9x9 for filtering line p_5 are not yet available and are replaced by pre-DF pixels. In Fig. 18, where cross9x9 is also selected for ALF, SAO output pixels of the last three rows (i.e., lines p_0-p_2) of the cross9x9 for filtering line p_4 are not yet available and are replaced by pre-DF pixels. As for filtering line p_3, it originally needs line q_0, which belongs to the lower LCU. In this case, the filter shape is reduced to cross9x7, and the coefficients at the two vertical ends of the original cross9x9 are added to the center pixel. In this way, it is unnecessary to change ALF syntax, and normalization of coefficients with multiplications and divisions can be avoided. Similarly,

SAO output pixels of the last three rows (i.e., lines p_0-p_2) of the cross9x7 for filtering line p_3 are not yet available and are replaced by pre-DF pixels. In Fig. 19, all pixels above the VB have been processed by ALF. At this moment, pixels above the VB can be written to a decoded picture buffer before the lower LCU comes.

[0079] Figs. 20-29 illustrate the proposed luma VB processing for pixels below the VB. In Fig. 20, four lines (i.e., p_0-p_3) of pre-DF pixels can be read from the DF line buffers. In Fig. 21, pixels of line p_3 and pixels below the VB are processed by horizontal DF. Please note that calculating horizontal DF decisions of lines p_0-p_3 requires pre-DF pixels of lines p_0-p_7. In order to save line buffers for lines p_4-p_7, these horizontal DF decisions are computed and stored in a decision line buffer during the horizontal DF for lines p_3-p_7 in Fig. 11. The decision buffer only requires one bit per 8x8 block and can be simply implemented as on-chip registers or SRAMs. In Fig. 22, pixels below the VB are processed by vertical DF. Please remember that vertical DF decisions use pre-DF pixels, so the vertical DF decisions at the horizontal LCU boundary have to be calculated before the horizontal DF is done in Fig. 21. In Fig. 23, pixels below the VB are going to be processed by SAO. Originally, SAO processing of line p_2 may need DF output pixels of line p_3. Since DF output pixels of line p_3 are already available, the SAO input pixels are not changed. In Fig. 24, all pixels below the VB (and above the next VB) have been processed by SAO. In Figs. 25-28, pixels below the VB are going to be processed by ALF. Originally, calculating ALF block properties of lines p_0-p_3 requires SAO output pixels of lines p_0-p_4. However, SAO output pixels of lines p_3-p_4 are not available any more. Therefore, SAO output pixels of line p_3 are replaced by the available DF output pixels, as shown in Fig. 25. In addition, repetitive padding in the vertical direction is applied to generate pixels of line p_4 from DF output pixels of line p_3. During filtering, SAO output pixels above the VB may also be originally needed. In these cases, SAO output pixels are replaced by DF output pixels of line p_3, and padding or reducing the filter shape may be applied. In Fig. 26, where snowflake5x5 is selected for ALF, for filtering line p_2, SAO output pixels of the second row (i.e., line p_3) of the snowflake5x5 are replaced by DF output pixels, and SAO output pixels of the first row (i.e., line p_4) of the snowflake5x5 are replaced by nearest DF output pixels of line p_3. Moreover, the filtering output pixels of line p_2 are further averaged with SAO output pixels of line p_2 to generate final ALF output pixels of line p_2. For filtering line p_1, SAO output pixels of the first row (i.e., line p_3) of the snowflake5x5 are replaced by DF output pixels. In Fig. 27, where cross9x9 is selected for ALF, for filtering line p_2, the filter shape is reduced to cross9x3 with adding the six discarded coefficients to the center pixel, and SAO output pixels of the first row (i.e., line p_3) of the cross9x3 are replaced by DF output pixels. For filtering line p_1, the filter shape is reduced to cross9x5 with adding the four discarded coefficients to the center pixel, and SAO output pixels of the first row (i.e., line p_3) of the cross9x5 are replaced by the DF output pixels. In Fig. 28, where cross9x9 is also selected for ALF, for filtering line p_0, the filter shape is reduced to cross9x7 with adding the two discarded coefficients to the center pixel; and SAO output pixels of the first row (i.e., line p_3) of the cross9x7 are replaced by DF output pixels. For filtering line q_0, the filter shape is unchanged, and SAO output pixels of the first row (i.e., line p_3) of the cross9x9 are replaced by DF output pixels. In this way, it is unnecessary to change ALF syntax, and normalization of coefficients with multiplications and divisions can be avoided. In Fig. 29, all pixels below the VB (and above the next VB) have been processed by ALF. At this moment, pixels below the VB (and above the next VB) can be written to a decoded picture buffer.

[0080] The design principles of luma VB processing are summarized in Table 2. In Fig. 11 and Fig. 21, it can be seen that line p_3 is processed by horizontal DF twice. This only happens in LCU-based processing, not in picture-based processing like HM. The redundant computation causes very minor impact in hardware because

the DF hardware has been already allocated and the DF is not the throughput bottleneck of the system. The redundant computation can be avoided by adding one line buffer to store H-DF pixels of line p_3; however, this is definitely not a good trade-off in hardware.

Table 2. Summary of the proposed luma VB processing

Operation	To-Be-Processed Line	Design Principle
SAO pixel classification	1st line above the VB	Use pre-DF pixels
ALF block property calculation	1st line above the VB	Copy upper block properties
ALF snowflake filtering	1st line above the VB	Use pre-DF pixels
ALF snowflake filtering	2nd line above the VB	Use pre-DF pixels
ALF cross filtering	1st line above the VB	Use pre-DF pixels and reduce filter size
ALF cross filtering	2nd line above the VB	Use pre-DF pixels
ALF cross filtering	3rd line above the VB	Use pre-DF pixels
ALF cross filtering	4th line above the VB	Use pre-DF pixels
SAO pixel classification	1st line below the VB	Remain the same
ALF block property calculation	1st, 2nd, and 3rd lines below the VB	Use DF output pixels and padding
ALF snowflake filtering	1st line below the VB	Use DF output pixels, padding, and averaging
ALF snowflake filtering	2nd line below the VB	Use DF output pixels
ALF cross filtering	1st line below the VB	Use DF output pixels and reduce filter size
ALF cross filtering	2nd line below the VB	Use DF output pixels and reduce filter size
ALF cross filtering	3rd line below the VB	Use DF output pixels and reduce filter size
ALF cross filtering	4th line below the VB	Use DF output pixels

[0081] The proposed chroma VB processing has the same concepts and similar principles as the proposed luma VB processing and is summarized in Table 3. Figs. 30-38 show the chroma VB processing for pixels above the VB, and Figs. 39-47 show the chroma VB processing for pixels below the VB. Please note that there is no redundant computation of chroma horizontal DF for line p_1 because the chroma vertical DF does not use any pre-DF pixel and the H-DF pixels of line p_1 are already stored in the line buffer.

Table 3. Summary of the proposed chroma VB processing

Operation	To-Be-Processed Line	Design Principle
SAO pixel classification	1st line above the VB	Use pre-DF pixels
ALF snowflake filtering	1st line above the VB	Use pre-DF pixels, padding, and averaging
ALF snowflake filtering	2nd line above the VB	Use pre-DF pixels
ALF cross filtering	1st line above the VB	Use pre-DF pixels and reduce filter size
ALF cross filtering	2nd line above the VB	Use pre-DF pixels and reduce filter size
ALF cross filtering	3rd line above the VB	Use pre-DF pixels and reduce filter size
ALF cross filtering	4th line above the VB	Use pre-DF pixels
SAO pixel classification	1st line below the VB	Remain the same
ALF snowflake filtering	1st line below the VB	Use DF output pixels, padding, and averaging
ALF snowflake filtering	2nd line below the VB	Use DF output pixels
ALF cross filtering	1st line below the VB	Use DF output pixels and reduce filter size
ALF cross filtering	2nd line below the VB	Use DF output pixels and reduce filter size
ALF cross filtering	3rd line below the VB	Use DF output pixels and reduce filter size
ALF cross filtering	4th line below the VB	Use DF output pixels

[0082] The concept of virtual boundaries can still be applied in VVC to reduce the line buffer usage. In the above description, largest coding unit (LCU) is renamed as coding tree unit (CTU) in VVC. However, some modifications are proposed to fit the features of current ALF design.

[0083] Method 1

[0084] The position of virtual boundaries should be aligned with the boundaries of smallest units (e.g., 4x4 blocks). Since all of pixels in one 4x4 luma block share the same filter coefficients (i.e., share the same ALF processing), the process of ALF with virtual boundaries becomes simple when virtual boundaries are aligned with the boundaries of 4x4 luma blocks. In other words, all of pixels in one 4x4 luma block should be located at the same side of virtual boundaries. Virtual boundaries cannot partition any 4x4 luma block into two regions. Since there is no classification for chroma component, no constraint related to the position of virtual boundaries for chroma component is required. In one embodiment, the space between the proposed virtual boundaries and the horizontal CTU boundaries is set to four pixels for luma (i.e. N=4 in Fig. 9A and Fig. 9B) and one pixel for chroma (i.e. N=1 in Fig. 9A and Fig. 9B).

[0085] Method 2

[0086] Since the block-based classification is used to achieve a high coding gain of ALF, those 4x4 blocks above the virtual boundaries or below the virtual boundaries can be treated as different groups compared to the other 4x4 blocks. As shown in Fig. 7, the virtual boundary is set as the boundary between line J and line K. The 4x4 blocks in lines G-J and lines K-N are treated as different groups comparted to 4x4 blocks in other lines, such as lines C-F. We can apply different classification rules to the 4x4 blocks in lines G-J and lines K-N. In other words, the classification rules used to group 4x4 blocks into different categories include the position of 4x4 blocks or virtual boundaries. In one embodiment, there are original 25 categories for normal 4x4 blocks; an additional (i.e., 26-th) category corresponding to the case that those 4x4 blocks being above the virtual boundaries, and another additional (i.e., 27-th) category corresponding to the case that those 4x4 blocks being below the virtual boundaries. In another embodiment, there are original 25 categories for normal 4x4 blocks; those 4x4 blocks above or below the virtual boundaries are classified into one additional 26-th category. In another embodiment, there are original 25 categories for normal 4x4 blocks; those 4x4 blocks above or below the virtual boundaries are classified into another additional 25 categories by using the same classification rules. In one embodiment, ALF is not applied to these 4x4 blocks above or below the virtual boundaries. In another embodiment, the decision of disabling ALF for these 4x4 blocks above or below the virtual boundaries is signalled at a sequence level, picture level, tile level or slice level.

[0087] For one 4x4 block, one 10x10 window is required to calculate the block property. In one embodiment, the required pixel at the other side of virtual boundaries are not used in the process of block property calculation. In another embodiment, the padding technology is used to generate the required data, including activity, gradient, gradient directions, and so on, when the required pixel are at the other side of virtual boundaries.

[0088] Method 3

[0089] The position of virtual boundaries is the middle position in the vertical direction of one 4x4 luma block. In one embodiment, the space between the proposed virtual boundaries and the horizontal CTU boundaries is set as six pixels for luma (i.e. N=6 in Fig. 9A and Fig. 9B) and one pixel for chroma (i.e. N=1 in Fig. 9A and Fig. 9B). Those 4x4 luma blocks cross virtual boundaries (e.g. 4x4 blocks in lines G-J in Fig. 7) are not filtered by ALF.

[0090] Method 4

[0091] The concept of virtual boundaries is further extended to be one guard band. For those pixels in the guard band, ALF is disabled. For those to-be-filtered pixels or to-be-filtered 4x4 blocks above the guard band,

the pixels below the guard band cannot be used but those pixels in the guard band can be referenced. For those to-be-filtered pixels or to-be-filtered 4x4 blocks below the guard band, the pixels above the guard band cannot be used but those pixels in the guard band can be referenced. In one embodiment, the guard band is set as four pixels for luma (i.e. N=6 in Fig. 9A and Fig. 9B) and the height is 4 pixels. That is, the guard band is defined as the region of lines G-J in Fig. 7. For those to-be-filtered pixels or to-be-filtered 4x4 blocks above the guard band (i.e., lines A-F in Fig. 7) only pixels above and including line J can be used. For those to-be-filtered pixels or to-be-filtered 4x4 blocks below the guard band (i.e., lines K-P in Fig. 7) only pixels below and including line G can be used. In another embodiment, for those to-be-filtered pixels or to-be-filtered 4x4 blocks below the guard band (i.e., lines K-P in Fig. 7) only pixels below and including some predefined line can be used and padding technology is used to generate the required pixels if the required pixel is above the predefined Line. For example, the predefined line can be line I or line J.

[0092] Method 5

[0093] When the concept of virtual boundaries or guard band is applied and padding technology is used to generate the required pixels instead of accessing the real pixels, the output value of ALF can be the weighted sum of filtered result and the unfiltered pixel. The weights for filtered results are decreased when less real pixels are used in the filtering process (i.e. more padded pixels are used in the filtering process). In one embodiment, the virtual boundaries disclosed in Method 1 is used. Therefore, the space between the proposed virtual boundaries and the horizontal CTU boundaries is set to four pixels for luma (i.e. N=4 in Fig. 9A and Fig. 9B) and one pixel for chroma (i.e. N=1 in Fig. 9A and Fig. 9B). The output value of ALF in line G and line N is equal to the filtered result. The output value of ALF in line H and line M is equal to ((the filtered result + the unfiltered result) / 2). The output value of ALF in line I and line L is equal to ((the filtered result + 3 x the unfiltered result) / 4). The output value of ALF in line J and line K is equal to ((the filtered result + 7 x the unfiltered result) / 8). In another embodiment, the output value of ALF in line G and line N is equal to ((the filtered result + the unfiltered result) / 2). The output value of ALF in Line H and Line M is equal to (the filtered result + 3 x the unfiltered result) / 4. The output value of ALF in Line I and Line L is equal to (the filtered result + 7 x the unfiltered result) / 8. The output value of ALF in line J and line K is equal to (the filtered result + 15 x the unfiltered result) / 16. In another embodiment, the output value of ALF in line G and line N is equal to (4 x the filtered result + 0 x the unfiltered result) / 4. The output value of ALF in line H and line M is equal to (3 x the filtered result + 1 x the unfiltered result) / 4. The output value of ALF in line I and line L is equal to ((2 x the filtered result + 2 x the unfiltered result) / 4). The output value of ALF in line J and line K is equal to ((1 x the filtered result + 3 x the unfiltered result) / 4). In another embodiment, the rounding offset is considered when generating the weighted sum of the filtered result and the unfiltered result. In another embodiment, the output value of ALF in line G and line N is equal to ((7 x the filtered result + 1 x the unfiltered result) / 8). The output value of ALF in line H and line M is equal to ((5 x the filtered result + 3 x the unfiltered result) / 8). The output value of ALF in line I and line L is equal to ((3 x the filtered result + 5 x the unfiltered result) / 8). The output value of ALF in line J and line K is equal to ((1 x the filtered result + 7 x the unfiltered result) / 8). In another embodiment, the rounding offset is considered when generating the weighted sum of the filtered result and the unfiltered result. In another embodiment, the rounding offset is considered when generating the weighted sum of the filtered result and the unfiltered result.

[0094] Method 6

[0095] The padding technology in the above methods can be direct padding, padding with even mirroring, padding with odd mirroring, some predefined value, to-be-filtered pixel, to-be-filtered pixel row, some intermediate result before ALF process, or some combination of the above. In another embodiment, the padding technology is to use the symmetric line in the filter shape when the required pixels are unavailable. For example, the virtual boundaries proposed in Method 1 is used. The space between the proposed virtual boundaries and the horizontal CTU boundaries is set to four pixels for luma (i.e. N=4 in Fig. 9A and Fig. 9B) and one pixel for chroma (i.e. N=1 in Fig. 9A and Fig. 9B). When the to-be-filtered pixels are located in line K, then the real pixels in line H-J cannot be accessed because of virtual boundaries. We use line L to replace line J, line M to replace line I, and line N to replace line K. When the to-be-filtered pixels are located in line L, then the real pixels in line I and line J cannot be accessed; line N is used to replace line J and line O is used to replace line I.

[0096] The above methods can be used individually or combined together to achieve one total solution in order to reduce line buffer usage of ALF.

[0097] Method 7

[0098] In one embodiment, the required pixels at the other side of virtual boundaries are replaced by the padded samples in ALF process, including classification and filtering operations. In other words, padding technology is used to generate the required samples for all processes in ALF, instead of accessing the real pixels at the other side of virtual boundaries. For example, when calculating gradients, if some of the required pixels in one 10x10 window used in classification are on the other side of virtual boundaries, then padding technology is used to generate the corresponding pixels.

[0099] In one embodiment, the operations of block classification are changed when part of the required pixels in one 10x10 window used in classification are on the other side of virtual boundaries. For example, if part of the required pixels in one 10x10 window used in classification are on the other side of virtual boundaries, then those gradients with position on the other side of virtual boundaries are set as zeros (i.e. not used), then the summation of gradients are normalized by the total number of available gradients. In another example, if part of the required pixels in one 10x10 window used in classification are at the other side of virtual boundaries, then the pattern of required gradients is changed to avoid using those pixels on the other side of virtual boundaries.

[00100] Any of the foregoing proposed methods can be implemented in encoders and/or decoders. For example, any of the proposed methods can be implemented in an in-loop filtering module of an encoder and/or a decoder. Alternatively, any of the proposed methods can be implemented as a circuit coupled to in-loop filtering module of the encoder and/or the decoder.

[00101] Fig. 48 illustrates a flowchart of an exemplary in-loop processing of reconstructed video according to an embodiment of the present invention. The steps shown in the flowchart may be implemented as program codes executable on one or more processors (e.g., one or more CPUs) at the encoder side. The steps shown in the flowchart may also be implemented based hardware such as one or more electronic devices or processors arranged to perform the steps in the flowchart. According to this method, reconstructed pixels associated with a current picture are received in step 4810, wherein the current picture is divided into multiple blocks for said in-loop processing of the reconstructed pixels. A virtual boundary for to-be-processed pixels in the current picture is determined in step 4820, wherein the virtual boundary is aligned with block boundaries and at least one to-be-processed pixel on a first side of the virtual boundary requires one or more second pixels on a second side of the virtual boundary. Said in-loop processing is modified in step 4830 if a target to-be-processed pixel

requires at least one second pixel from the second side of the virtual boundary, wherein said modified in-loop processing eliminates a need for any second pixel on the second side of the virtual boundary.

[00102] Fig. 49 illustrates a flowchart of exemplary adaptive-loop-filter processing of reconstructed video according to an embodiment of the present invention. According to another method, reconstructed pixels associated with a current picture are received in step 4910, wherein the current picture is divided into multiple blocks for said adaptive-loop-filter processing of the reconstructed pixels. A virtual boundary for to-be-processed pixels in the current picture is determined in step 4920, wherein at least one to-be-processed pixel on a first side of the virtual boundary requires one or more second pixels on a second side of the virtual boundary. An extended block covering the target block is determined in step 4930 for a target block belonging to the to-be-processed pixels. Block classification is derived for the target block based on the extended block in step 4940, wherein if any target pixel of the extended block required for said deriving the block classification is on the second side of the virtual boundary, at least one operation associated with the block classification is modified. Said adaptive-loop-filter processing is applied to the target block using the block classification with said at least one operation modified in step 4950 if any target pixel of the extended block required for said deriving the block classification is on the second side of the virtual boundary.

[00103] The flowcharts shown are intended to illustrate an example of video coding according to the present invention. A person skilled in the art may modify each step, re-arranges the steps, split a step, or combine steps to practice the present invention without departing from the spirit of the present invention. In the disclosure, specific syntax and semantics have been used to illustrate examples to implement embodiments of the present invention. A skilled person may practice the present invention by substituting the syntax and semantics with equivalent syntax and semantics without departing from the spirit of the present invention.

[00104] The above description is presented to enable a person of ordinary skill in the art to practice the present invention as provided in the context of a particular application and its requirement. Various modifications to the described embodiments will be apparent to those with skill in the art, and the general principles defined herein may be applied to other embodiments. Therefore, the present invention is not intended to be limited to the particular embodiments shown and described, but is to be accorded the widest scope consistent with the principles and novel features herein disclosed. In the above detailed description, various specific details are illustrated in order to provide a thorough understanding of the present invention. Nevertheless, it will be understood by those skilled in the art that the present invention may be practiced.

[00105] Embodiment of the present invention as described above may be implemented in various hardware, software codes, or a combination of both. For example, an embodiment of the present invention can be one or more circuit circuits integrated into a video compression chip or program code integrated into video compression software to perform the processing described herein. An embodiment of the present invention may also be program code to be executed on a Digital Signal Processor (DSP) to perform the processing described herein. The invention may also involve a number of functions to be performed by a computer processor, a digital signal processor, a microprocessor, or field programmable gate array (FPGA). These processors can be configured to perform particular tasks according to the invention, by executing machine-readable software code or firmware code that defines the particular methods embodied by the invention. The software code or firmware code may be developed in different programming languages and different formats or styles. The software code may also be compiled for different target platforms. However, different code formats, styles and languages of

software codes and other means of configuring code to perform the tasks in accordance with the invention will not depart from the spirit and scope of the invention.

[00106] The invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described examples are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

CLAIMS

1. A method for in-loop processing of reconstructed video, the method comprising:

receiving reconstructed pixels associated with a current picture, wherein the current picture is divided into multiple blocks for said in-loop processing of the reconstructed pixels;

determining a virtual boundary for to-be-processed pixels in the current picture, wherein the virtual boundary is aligned with block boundaries and at least one to-be-processed pixel on a first side of the virtual boundary requires one or more second pixels on a second side of the virtual boundary; and

modifying said in-loop processing if a target to-be-processed pixel requires at least one second pixel from the second side of the virtual boundary, wherein said modified in-loop processing eliminates a need for any second pixel on the second side of the virtual boundary.

2. The method of Claim 1, wherein the multiple blocks correspond to 4x4 blocks.

3. The method of Claim 1, wherein the multiple blocks correspond to smallest units, and wherein all samples in each smallest blocks share an identical in-loop processing .

4. The method of Claim 1, wherein the in-loop processing corresponds to SAO (Sample Adaptive Offset) processing or ALF (Adaptive Loop Filter) processing.

5. The method of Claim 1, wherein the reconstructed pixels correspond to luma pixels.

6. The method of Claim 1, wherein for chroma reconstructed pixels of the current picture, a chroma virtual boundary is allowed to cross chroma blocks.

7. The method of Claim 1, wherein if the target to-be-processed pixel requires said at least one second pixel on the second side of the virtual boundary, said at least one second pixel from the second side of the virtual boundary are padded.

8. The method of Claim 1, wherein if the target to-be-processed pixel requires said at least one second pixel from the second side of the virtual boundary, said at least one second pixel from the second side of the virtual boundary are padded using symmetric padding.

9. The method of Claim 1, wherein said modified in-loop processing uses a modified filter with reduced footprint to eliminate the need for any second pixel on the second side of the virtual boundary.

10. The method of Claim 1, wherein said modified in-loop processing is applied to the reconstructed pixels at an encoder side.

11. The method of Claim 1, wherein said modified in-loop processing is applied to the reconstructed pixels at a decoder side.

12. An apparatus for in-loop processing of reconstructed video, the apparatus comprising one or more electronic circuits or processors arranged to:

receive reconstructed pixels associated with a current picture, wherein the current picture is divided into multiple blocks for said in-loop processing of the reconstructed pixels;

determine a virtual boundary for to-be-processed pixels in the current picture, wherein the virtual boundary is aligned with block boundaries and at least one to-be-processed pixel on a first side of the virtual boundary requires one or more second pixels on a second side of the virtual boundary; and

modify said in-loop processing if a target to-be-processed pixel requires at least one second pixel from the second side of the virtual boundary, wherein said modified in-loop processing eliminates a need for any second

pixel on the second side of the virtual boundary.

13. A method for adaptive-loop-filter processing of reconstructed video, the method comprising:

receiving reconstructed pixels associated with a current picture, wherein the current picture is divided into multiple blocks for said adaptive-loop-filter processing of the reconstructed pixels;

determining a virtual boundary for to-be-processed pixels in the current picture, wherein at least one to-be-processed pixel on a first side of the virtual boundary requires one or more second pixels on a second side of the virtual boundary;

for a target block belonging to the to-be-processed pixels, determining an extended block covering the target block;

deriving block classification for the target block based on the extended block, wherein if any target pixel of the extended block required for said deriving the block classification is on the second side of the virtual boundary, at least one operation associated with the block classification is modified; and

applying said adaptive-loop-filter processing to the target block using the block classification with said at least one operation modified if any target pixel of the extended block required for said deriving the block classification is on the second side of the virtual boundary.

14. The method of Claim 13, wherein an evaluation block is determined for the target block, wherein the target block is located at a center of the evaluation block and surrounding pixels are added to the evaluation block to form the extended block.

15. The method of Claim 14, wherein the target block corresponds to a 4x4 block, the evaluation block corresponds to an 8x8 block and the extended block corresponds to a 10x10 block.

16. The method of Claim 15, wherein the target block corresponds to a smallest unit, and wherein all samples in each smallest block share an identical adaptive-loop-filter processing.

17. The method of Claim 14, wherein the block classification is derived based on gradients at selected locations of the evaluation block.

18. The method of Claim 17, wherein if a target gradient for a target selected location of the evaluation block requires any target pixel of the extended block on the second side of the virtual boundary, the target gradient is excluded from a summation of gradients for the block classification or is set to zero.

19. The method of Claim 18, wherein the summation of gradients are weighted according to a total number of available gradients at the selected locations of the evaluation block.

20. The method of Claim 18, wherein the summation of gradients are normalized by a total number of available gradients at the selected locations of the evaluation block.

21. The method of Claim 17, wherein if a target gradient for a target selected location of the evaluation block requires any target pixel of the extended block on the second side of the virtual boundary, the target gradient for the target selected location of the evaluation block is modified to exclude said any target pixel of the extended block on the second side of the virtual boundary.

22. The method of Claim 17, wherein the selected locations of the evaluation block correspond to 2:1 horizontal sub-sampled locations with one-pixel offset between even lines and odd lines.

23. An apparatus for adaptive-loop-filter processing of reconstructed video, the apparatus comprising one or more electronic circuits or processors arranged to:

receive reconstructed pixels associated with a current picture, wherein the current picture is divided into

multiple blocks for said adaptive-loop-filter processing of the reconstructed pixels;

determine a virtual boundary for to-be-processed pixels in the current picture, wherein at least one to-be-processed pixel on a first side of the virtual boundary requires one or more second pixels on a second side of the virtual boundary;

for a target block belonging to the to-be-processed pixels, determine an extended block covering the target block;

derive block classification for the target block based on the extended block, wherein if any target pixel of the extended block required to derive the block classification is on the second side of the virtual boundary, at least one operation associated with the block classification is modified; and

apply said adaptive-loop-filter processing to the target block using the block classification with said at least one operation modified if any target pixel of the extended block required to derive the block classification is on the second side of the virtual boundary.

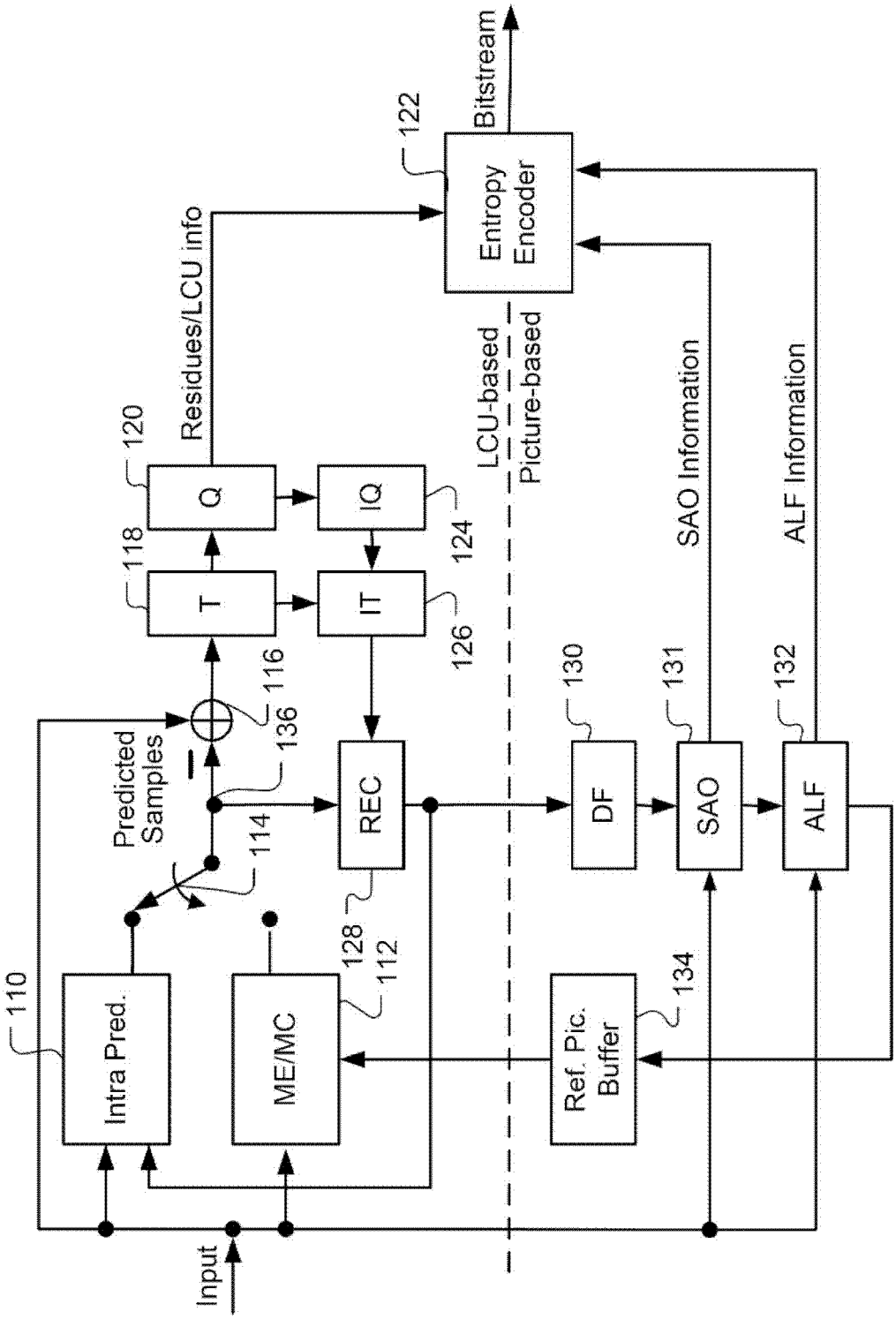
Bitstream
122
Entropy Encoder
Residues/LCU info
120
Q
IQ
124
118
T
IT
126
LCU-based
Picture-based
SAO Information
ALF Information
116
136
130
131
132
Predicted Samples
114
REC
DF
SAO
ALF
128
112
134
110
Intra Pred.
ME/MC
Ref. Pic. Buffer
Input

Fig. 1A

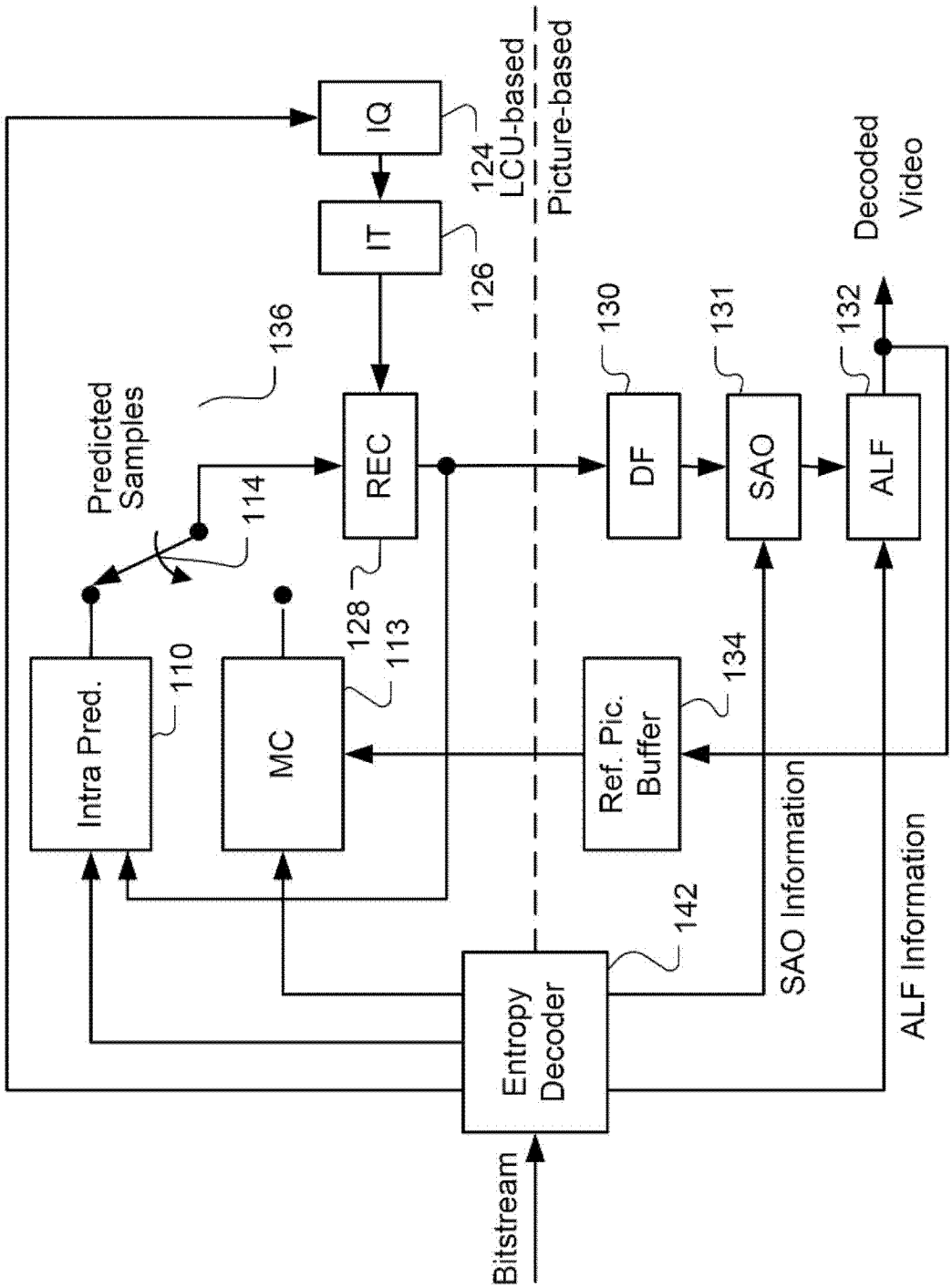
Bitstream
Entropy Decoder
142
Intra Pred.
110
MC
113
128
Predicted Samples
114
136
IQ
IT
126
124
LCU-based
Picture-based
REC
DF
130
SAO
131
ALF
132
Decoded Video
Ref. Pic. Buffer
134
SAO Information
ALF Information

Fig. 1B

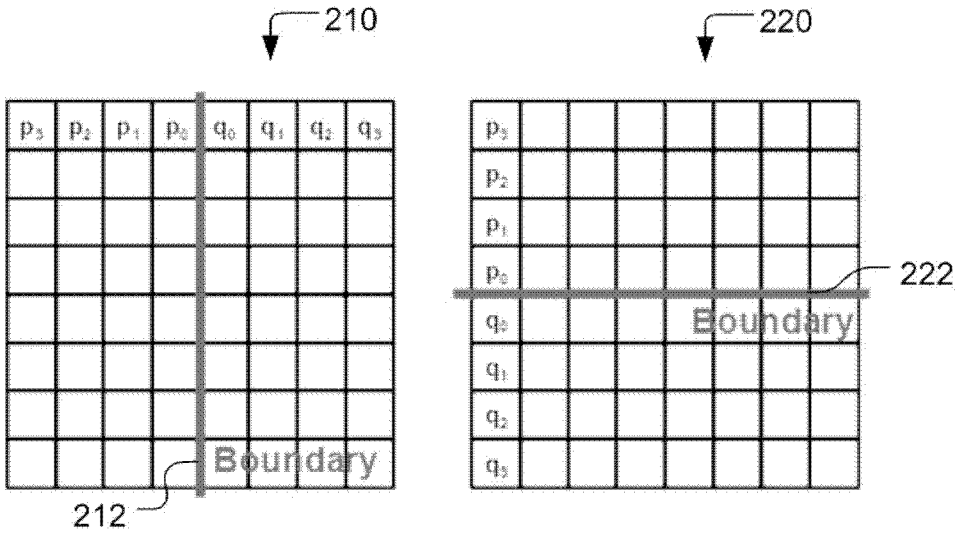

Fig. 2

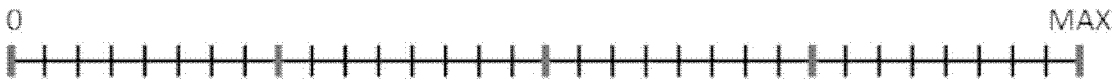

Fig. 3

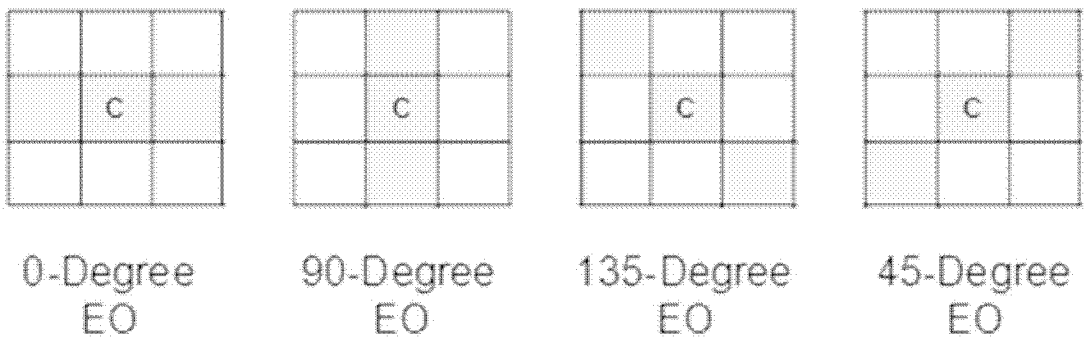

Fig. 4

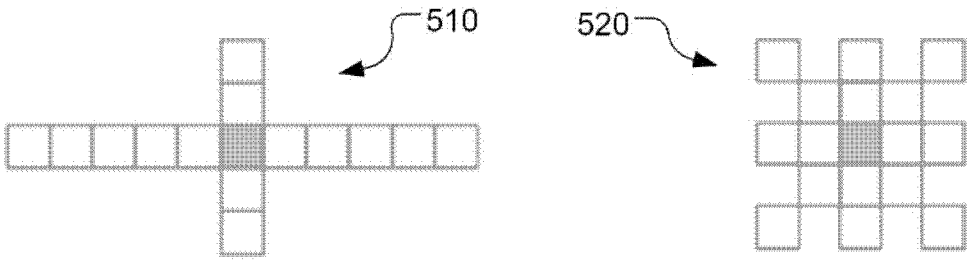

Fig. 5

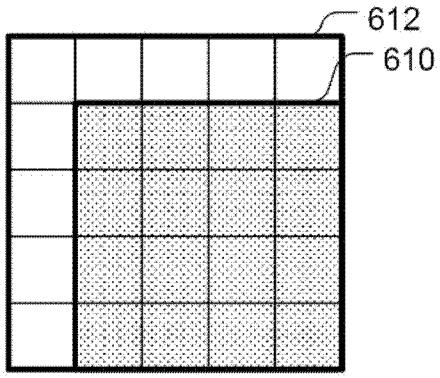

Fig. 6A

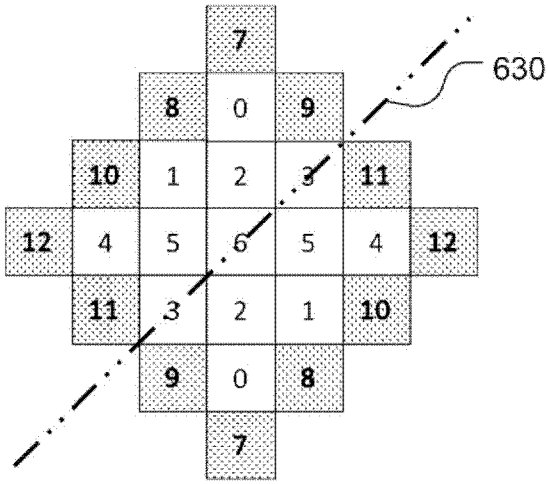

Fig. 6C

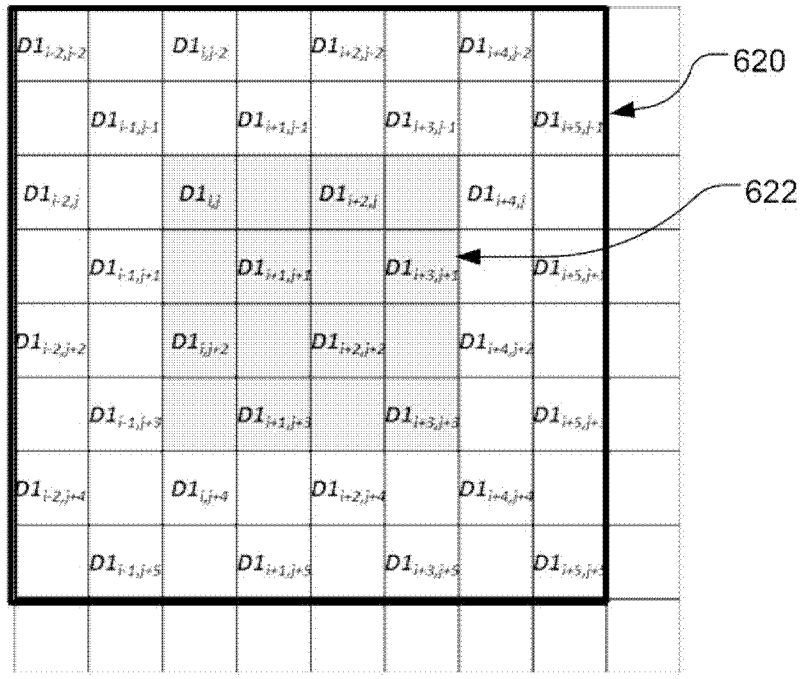

Fig. 6B

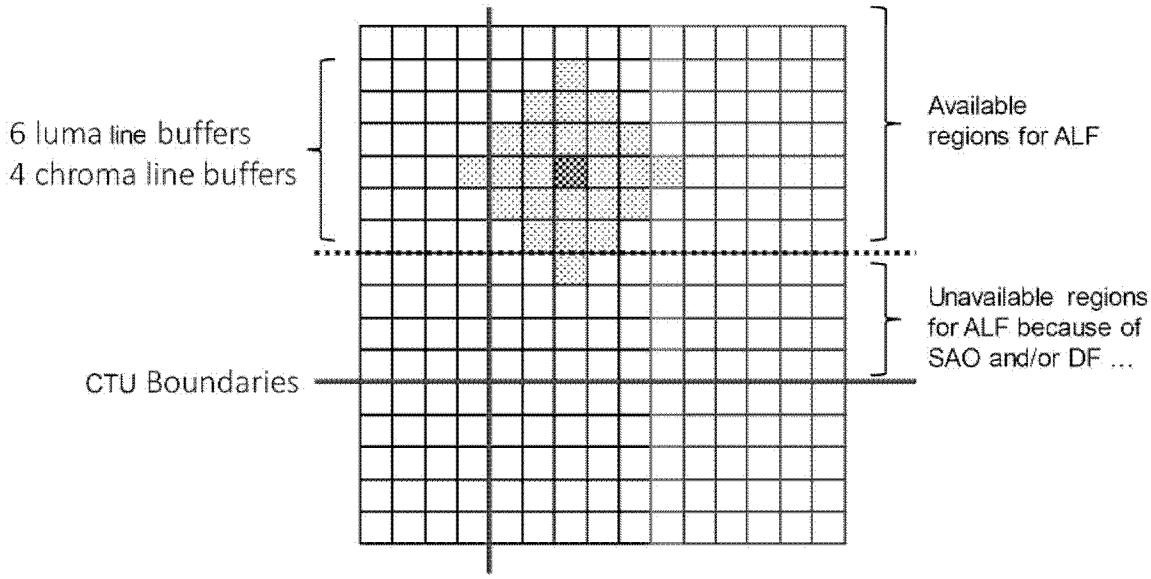
6 luma line buffers
4 chroma line buffers
CTU Boundaries
Available regions for ALF
Unavailable regions for ALF because of SAO and/or DF ...

Fig. 6D

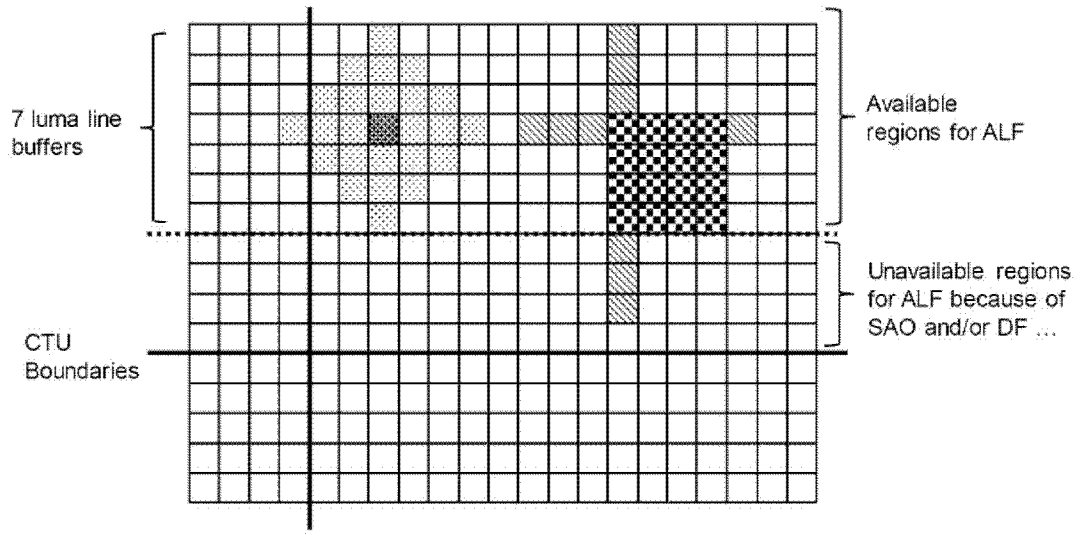
7 luma line buffers
CTU Boundaries
Available regions for ALF
Unavailable regions for ALF because of SAO and/or DF ...

Fig. 6E

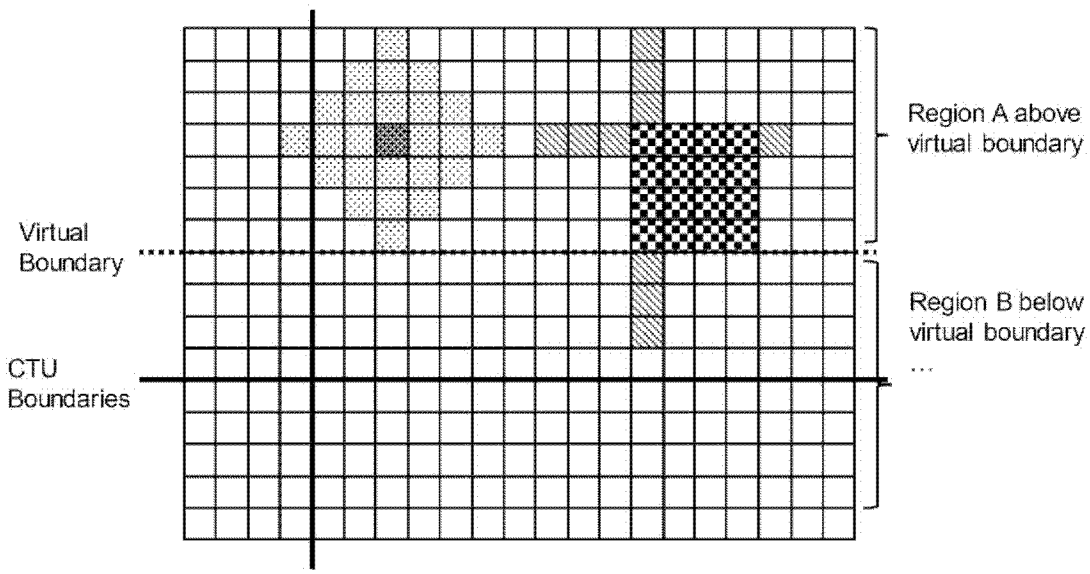
Region A above virtual boundary
Virtual Boundary
Region B below virtual boundary
...
CTU Boundaries

Fig. 6F

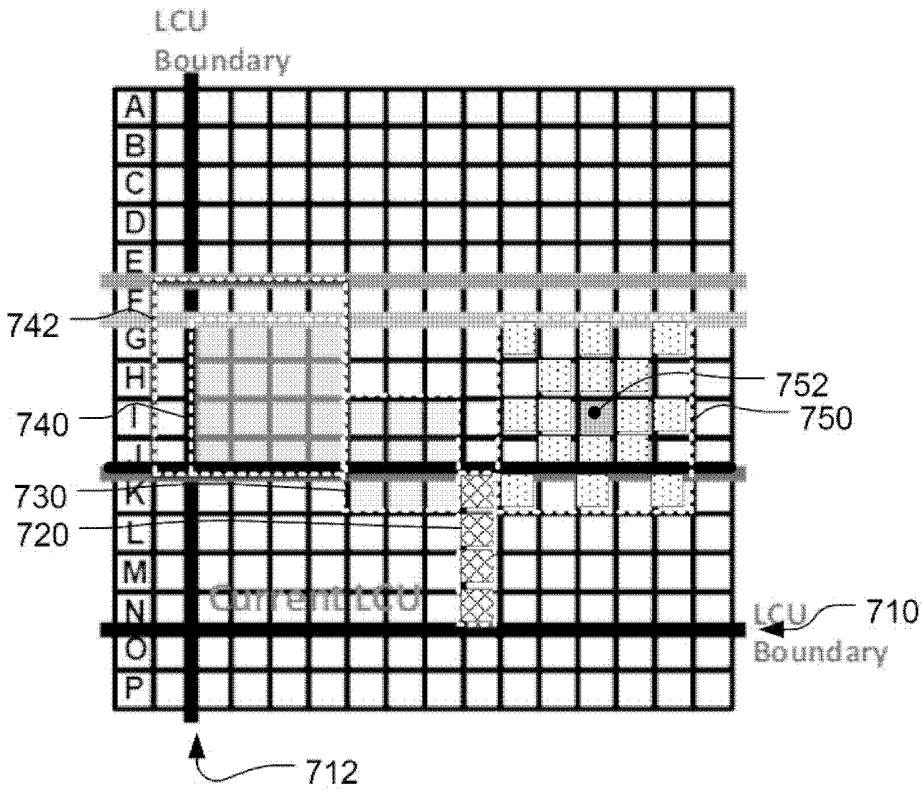
LCU
Boundary
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
742
740
730
720
752
750
Current LCU
LCU
Boundary
710
712

Fig. 7

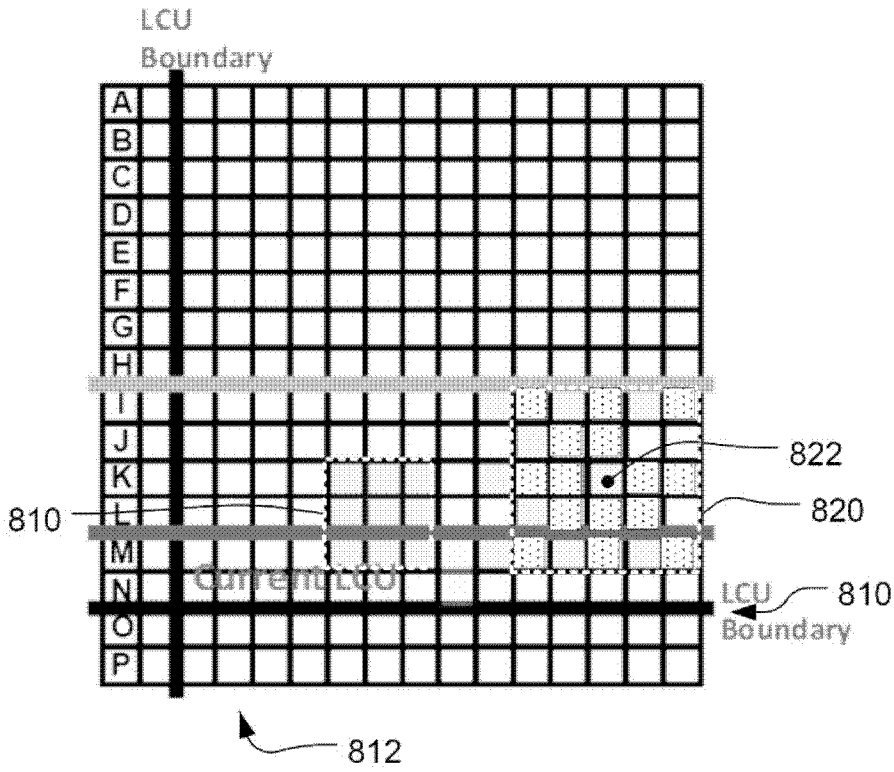
LCU
Boundary
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
822
820
810
Current LCU
LCU
Boundary
810
812

Fig. 8

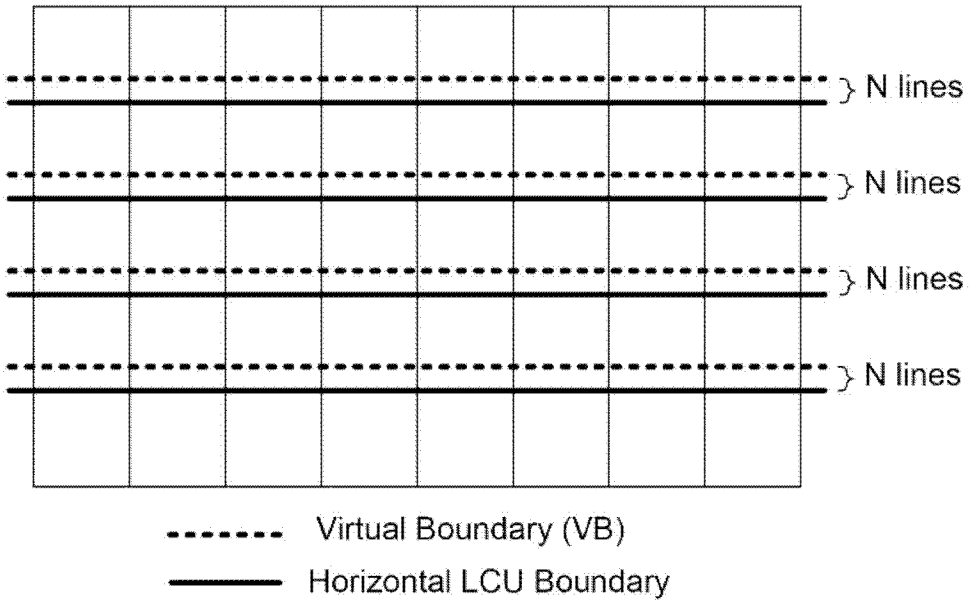
N lines
N lines
N lines
N lines
Virtual Boundary (VB)
Horizontal LCU Boundary

Fig. 9A

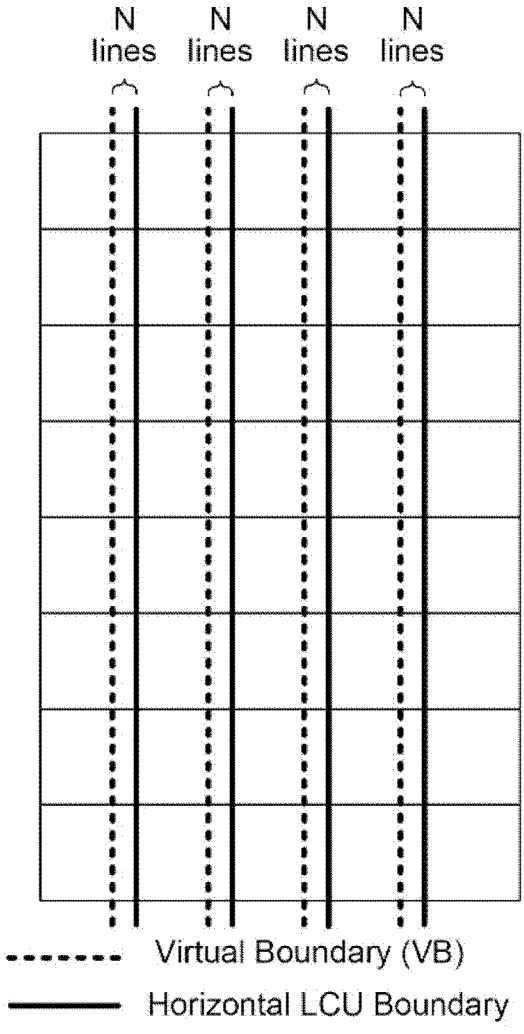
N lines
N lines
N lines
N lines
Virtual Boundary (VB)
Horizontal LCU Boundary

Fig. 9B

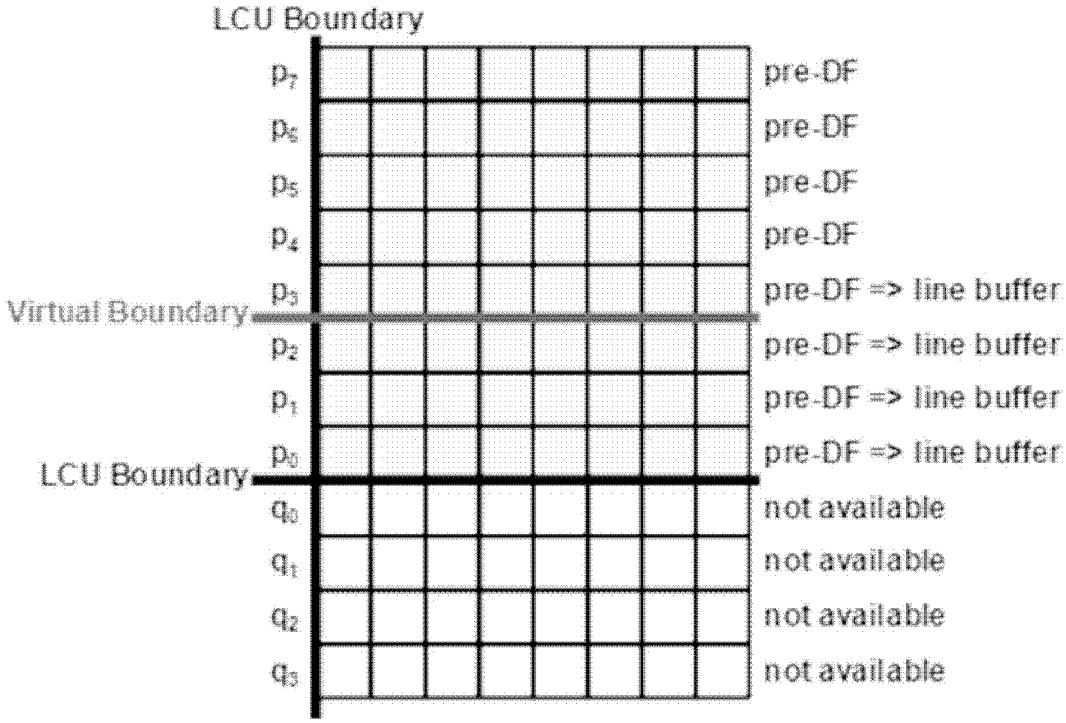
LCU Boundary
p_7
p_6
p_5
p_4
p_3
Virtual Boundary
p_2
p_1
p_0
LCU Boundary
q_0
q_1
q_2
q_3
pre-DF
pre-DF
pre-DF
pre-DF
pre-DF => line buffer
pre-DF => line buffer
pre-DF => line buffer
pre-DF => line buffer
not available
not available
not available
not available

Fig. 10

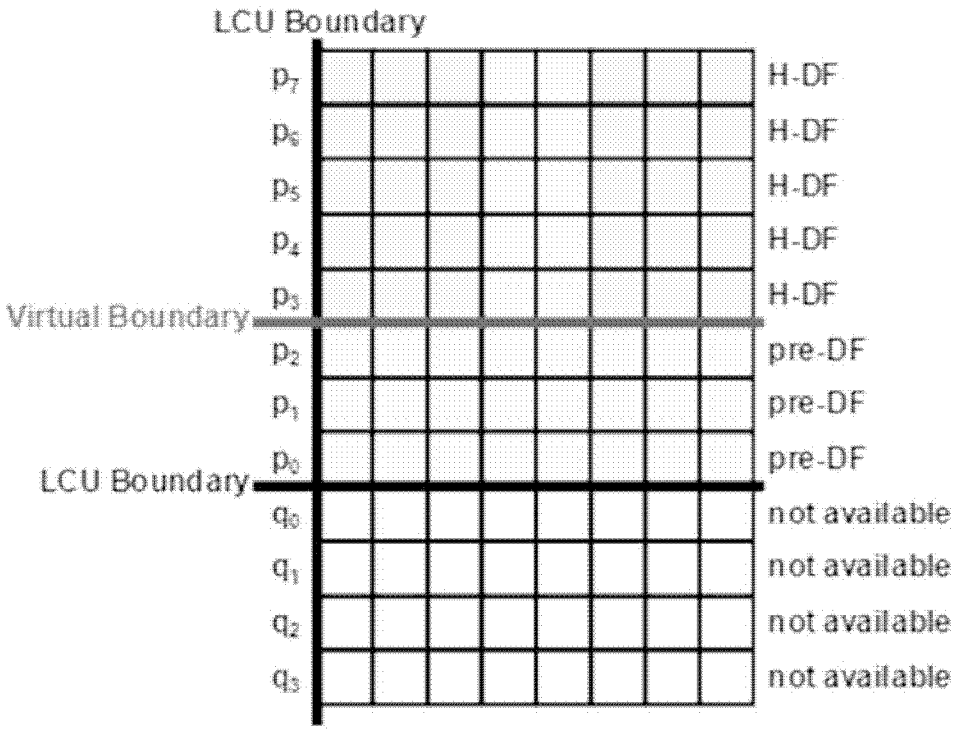
LCU Boundary
p_7
p_6
p_5
p_4
p_3
Virtual Boundary
p_2
p_1
p_0
LCU Boundary
q_0
q_1
q_2
q_3
H-DF
H-DF
H-DF
H-DF
H-DF
pre-DF
pre-DF
pre-DF
not available
not available
not available
not available

Fig. 11

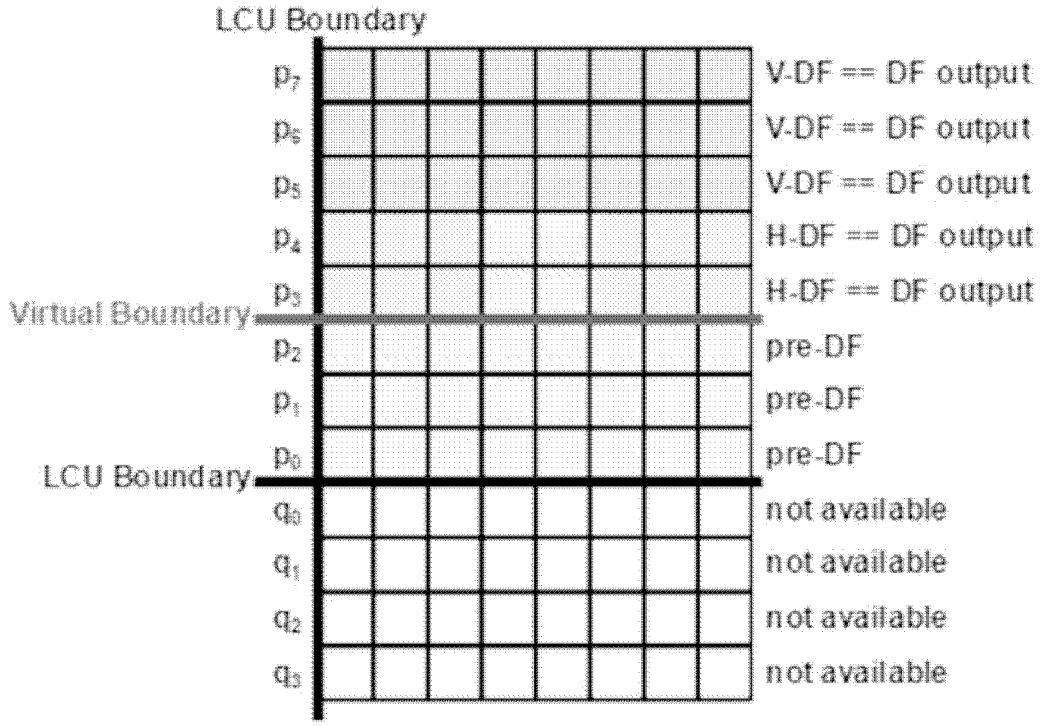
LCU Boundary
p7
p6
p5
p4
p3
Virtual Boundary
p2
p1
p0
LCU Boundary
q0
q1
q2
q3
V-DF == DF output
V-DF == DF output
V-DF == DF output
H-DF == DF output
H-DF == DF output
pre-DF
pre-DF
pre-DF
not available
not available
not available
not available

Fig. 12

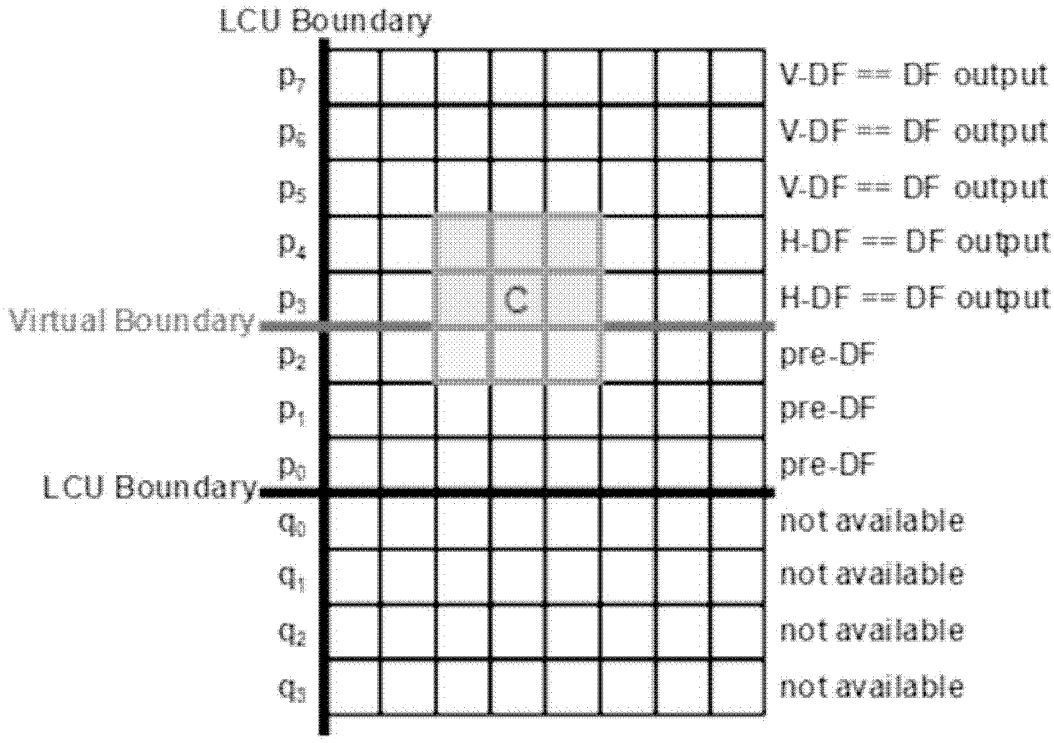
LCU Boundary
p7
p6
p5
p4
p3
C
Virtual Boundary
p2
p1
p0
LCU Boundary
q0
q1
q2
q3
V-DF == DF output
V-DF == DF output
V-DF == DF output
H-DF == DF output
H-DF == DF output
pre-DF
pre-DF
pre-DF
not available
not available
not available
not available

Fig. 13

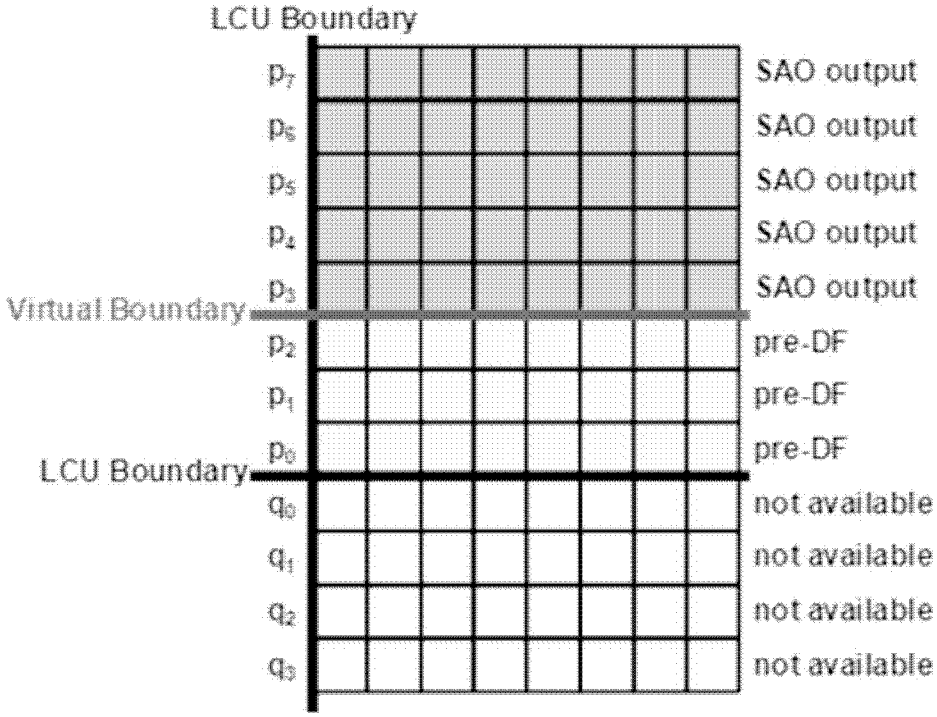
LCU Boundary
p_7
p_6
p_5
p_4
p_3
Virtual Boundary
p_2
p_1
p_0
LCU Boundary
q_0
q_1
q_2
q_3
SAO output
SAO output
SAO output
SAO output
SAO output
pre-DF
pre-DF
pre-DF
not available
not available
not available
not available

Fig. 14

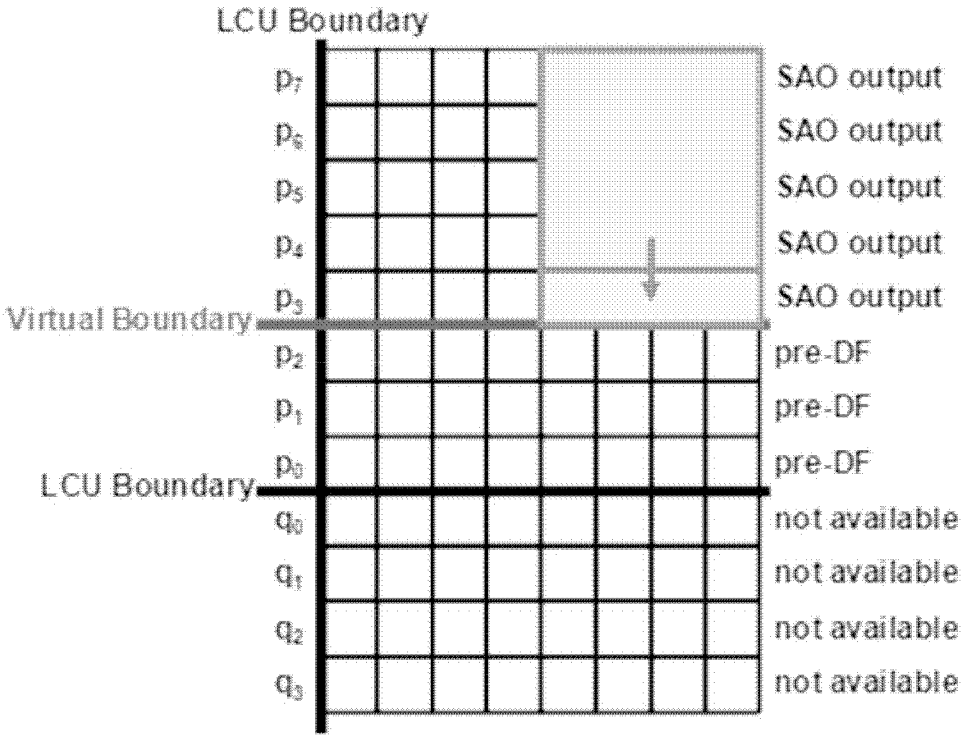
LCU Boundary
p_7
p_6
p_5
p_4
p_3
Virtual Boundary
p_2
p_1
p_0
LCU Boundary
q_0
q_1
q_2
q_3
SAO output
SAO output
SAO output
SAO output
SAO output
pre-DF
pre-DF
pre-DF
not available
not available
not available
not available

Fig. 15

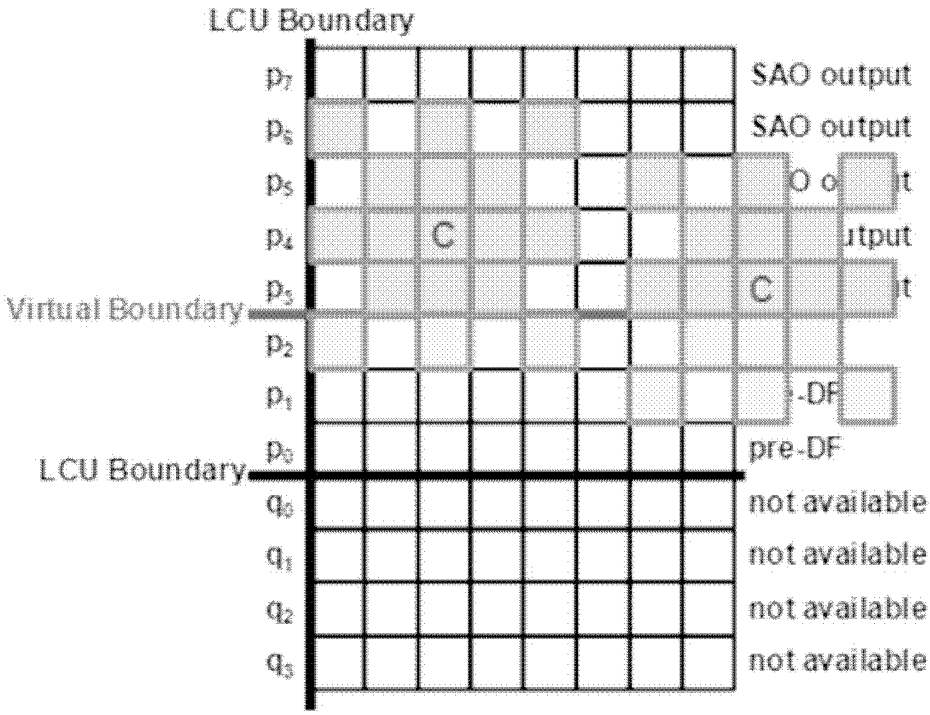
LCU Boundary
p7
p6
p5
p4
p3
Virtual Boundary
p2
p1
p0
LCU Boundary
q0
q1
q2
q3
SAO output
SAO output
C
C
pre-DF
not available
not available
not available
not available

Fig. 16

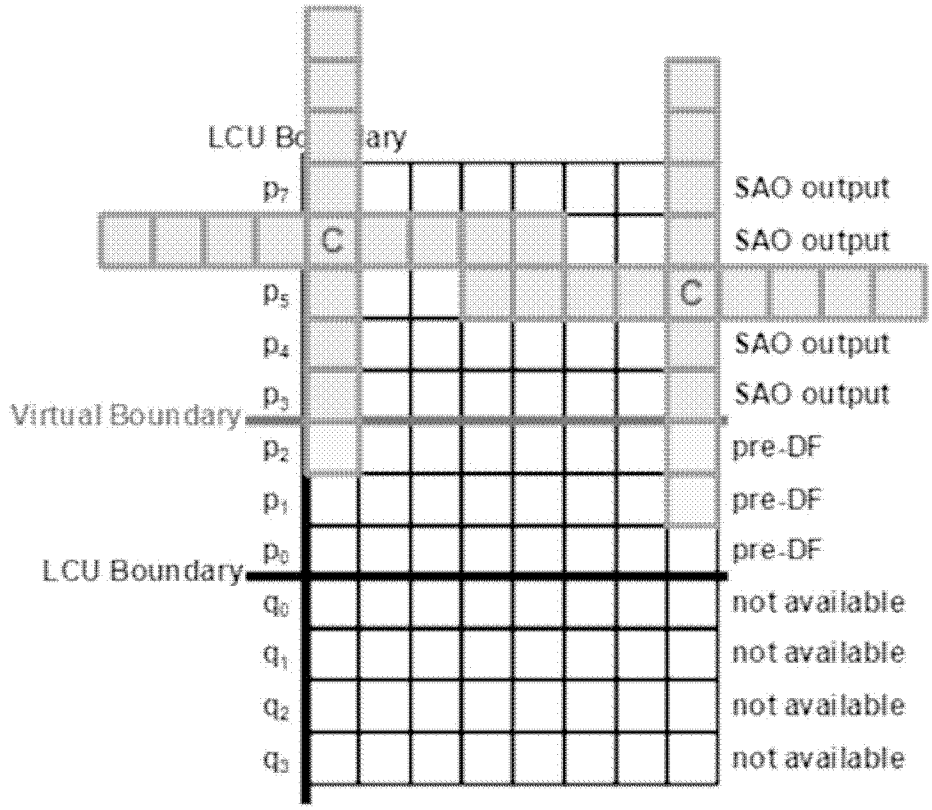
p7
C
p5
C
p4
p3
Virtual Boundary
p2
p1
p0
LCU Boundary
q0
q1
q2
q3
SAO output
SAO output
SAO output
SAO output
pre-DF
pre-DF
pre-DF
not available
not available
not available
not available

Fig. 17

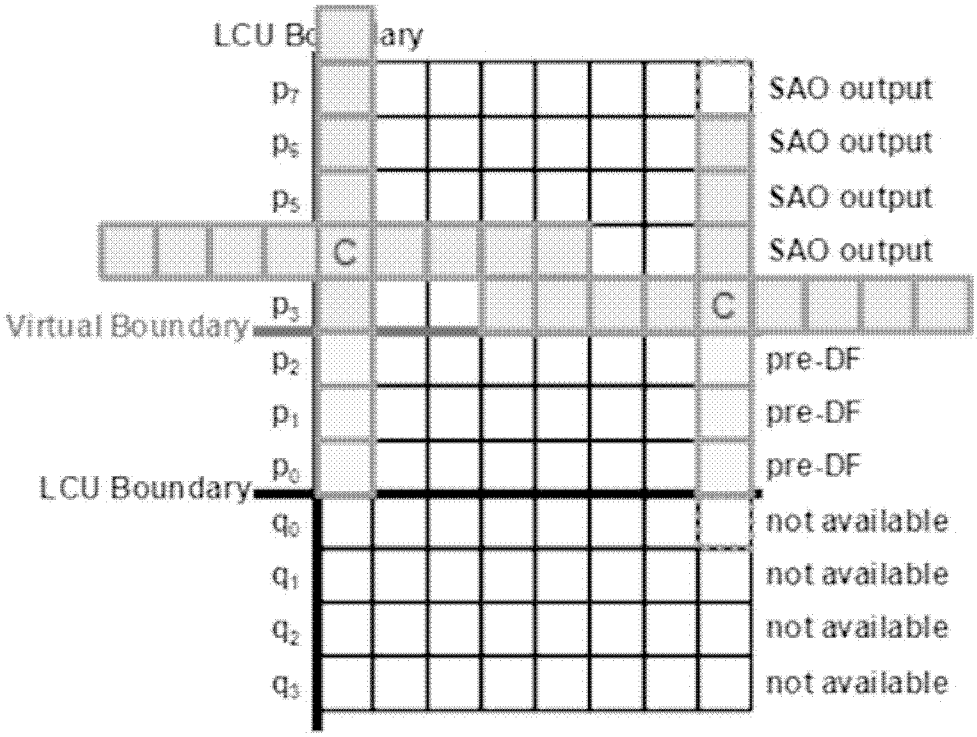
LCU Boundary
p7
p6
p5
C
p3
Virtual Boundary
p2
p1
p0
LCU Boundary
q0
q1
q2
q3
SAO output
SAO output
SAO output
SAO output
C
pre-DF
pre-DF
pre-DF
not available
not available
not available
not available

Fig. 18

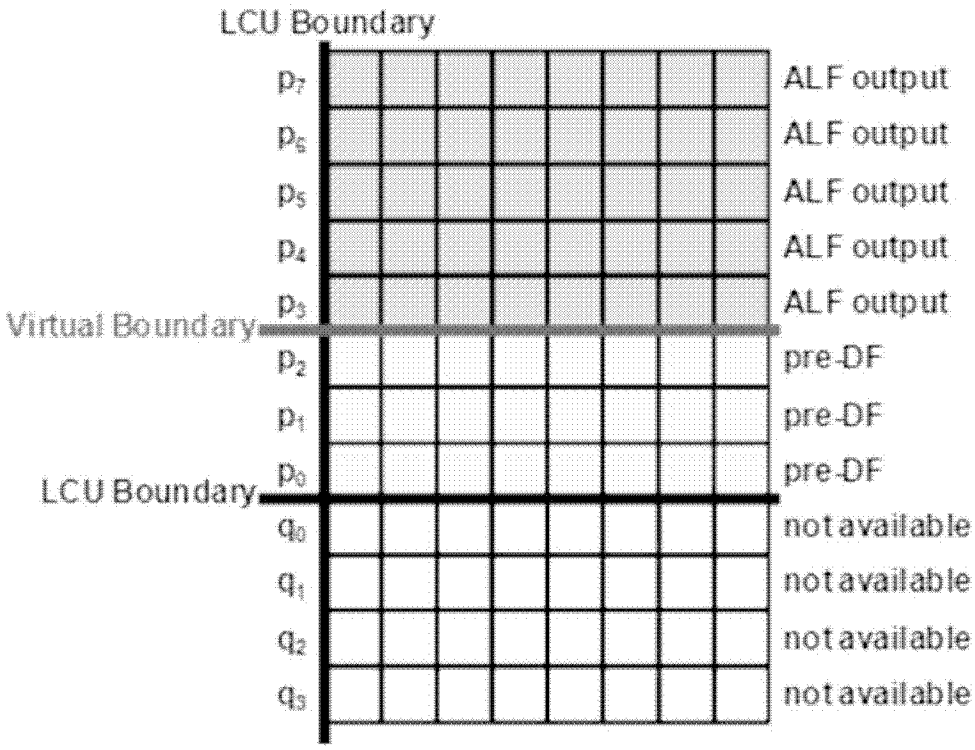
LCU Boundary
p7
p6
p5
p4
p3
Virtual Boundary
p2
p1
p0
LCU Boundary
q0
q1
q2
q3
ALF output
ALF output
ALF output
ALF output
ALF output
pre-DF
pre-DF
pre-DF
not available
not available
not available
not available

Fig. 19

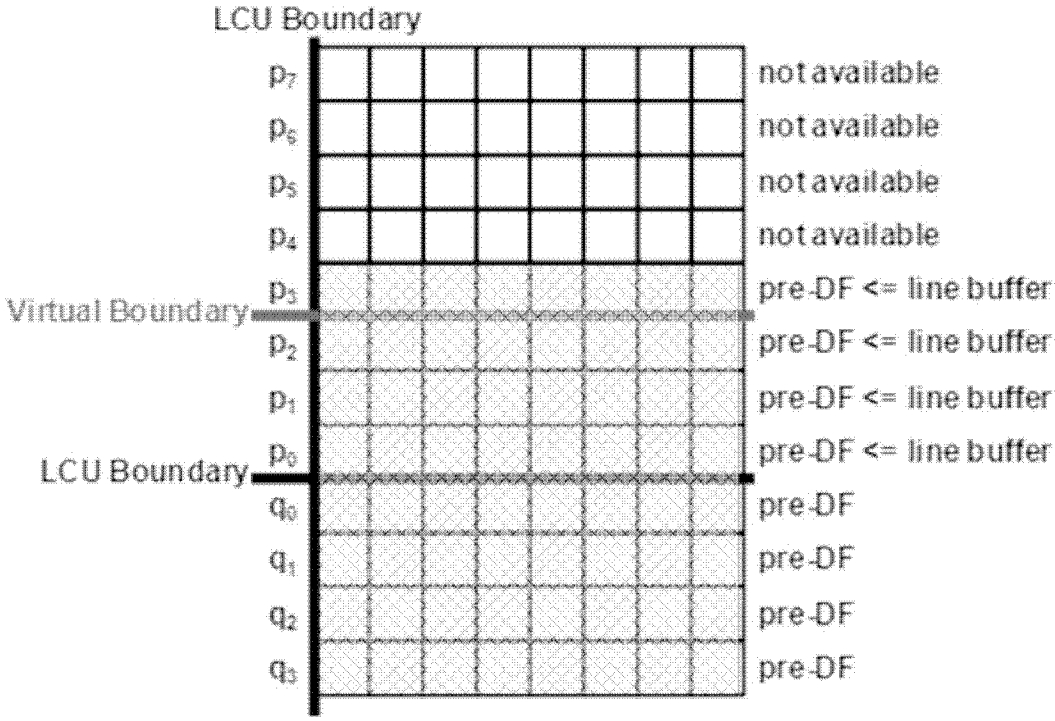
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
p0
q0
q1
q2
q3
Virtual Boundary
LCU Boundary
not available
not available
not available
not available
pre-DF <= line buffer
pre-DF <= line buffer
pre-DF <= line buffer
pre-DF <= line buffer
pre-DF
pre-DF
pre-DF
pre-DF

Fig. 20

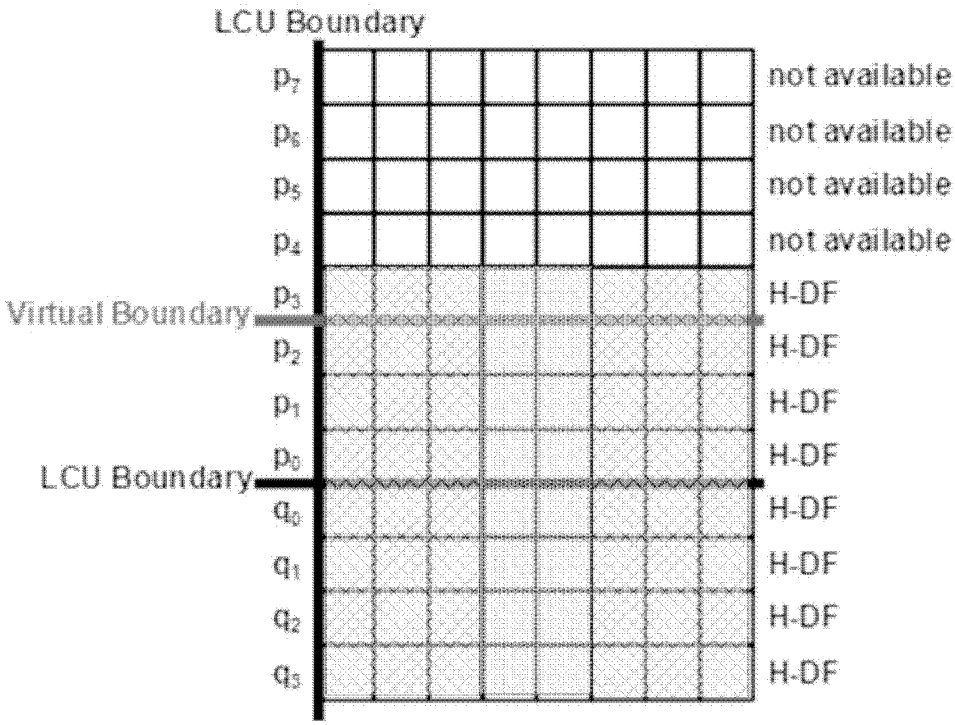
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
p0
q0
q1
q2
q3
Virtual Boundary
LCU Boundary
not available
not available
not available
not available
H-DF
H-DF
H-DF
H-DF
H-DF
H-DF
H-DF
H-DF

Fig. 21

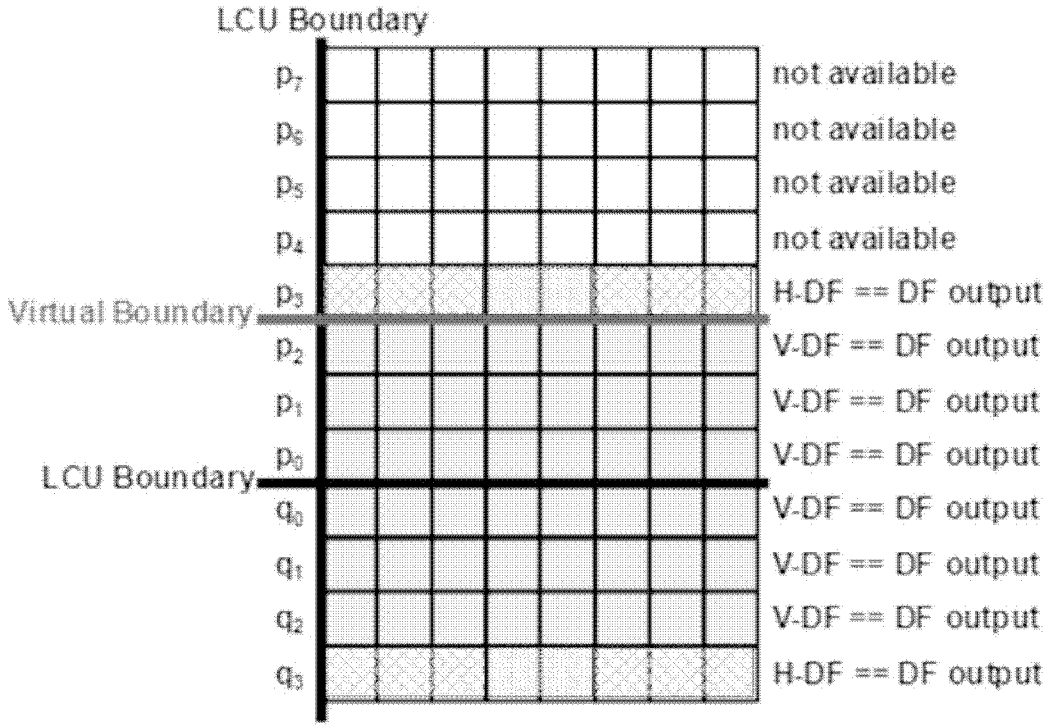
LCU Boundary
p7
p6
p5
p4
p3
Virtual Boundary
p2
p1
p0
LCU Boundary
q0
q1
q2
q3
not available
not available
not available
not available
H-DF == DF output
V-DF == DF output
V-DF == DF output
V-DF == DF output
V-DF == DF output
V-DF == DF output
V-DF == DF output
H-DF == DF output

Fig. 22

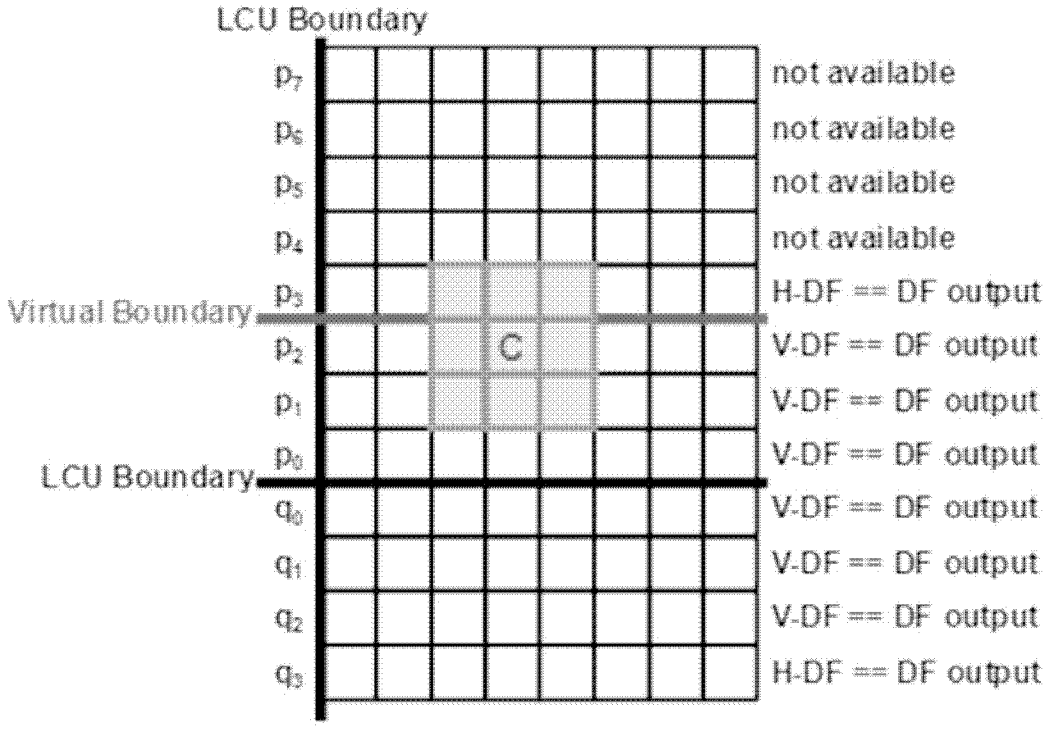
LCU Boundary
p7
p6
p5
p4
p3
Virtual Boundary
p2
p1
p0
LCU Boundary
q0
q1
q2
q3
C
not available
not available
not available
not available
H-DF == DF output
V-DF == DF output
V-DF == DF output
V-DF == DF output
V-DF == DF output
V-DF == DF output
V-DF == DF output
H-DF == DF output

Fig. 23

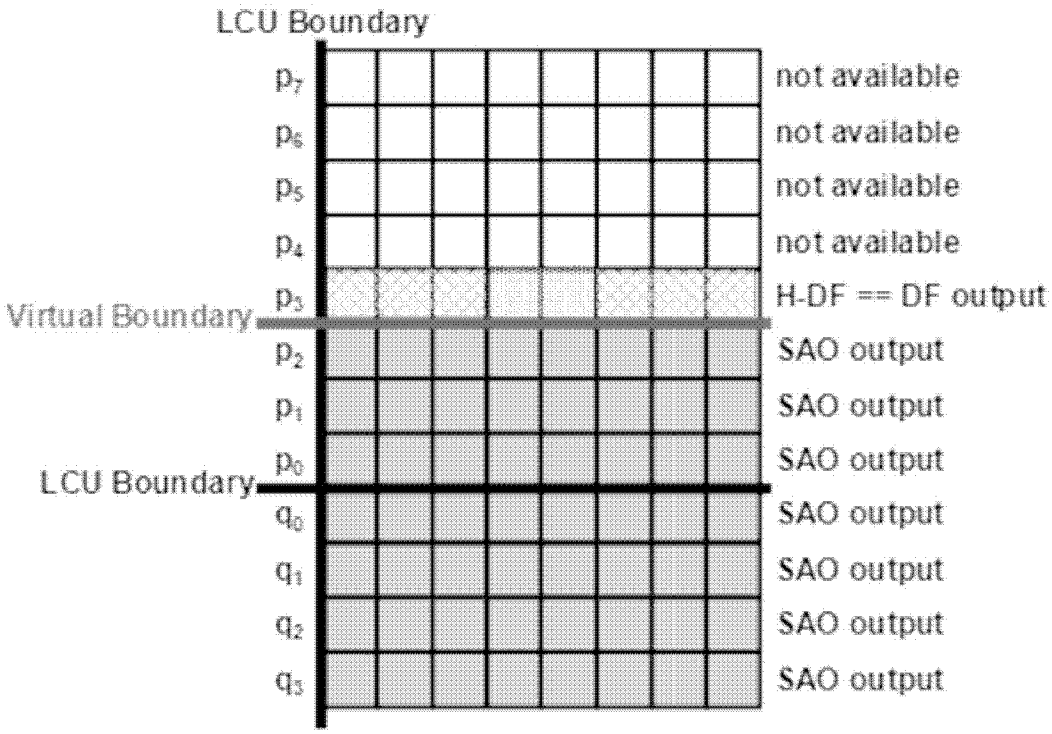
LCU Boundary
p7
p6
p5
p4
p3
Virtual Boundary
p2
p1
p0
LCU Boundary
q0
q1
q2
q3
not available
not available
not available
not available
H-DF == DF output
SAO output
SAO output
SAO output
SAO output
SAO output
SAO output
SAO output

Fig. 24

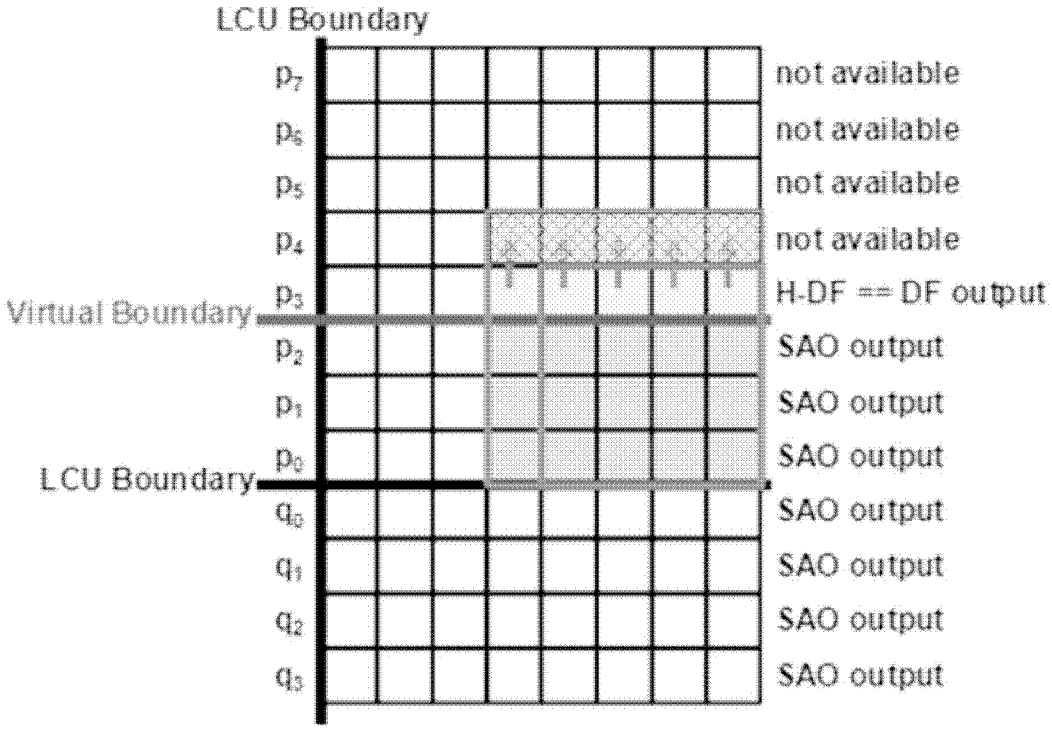
LCU Boundary
p7
p6
p5
p4
p3
Virtual Boundary
p2
p1
p0
LCU Boundary
q0
q1
q2
q3
not available
not available
not available
not available
H-DF == DF output
SAO output
SAO output
SAO output
SAO output
SAO output
SAO output
SAO output

Fig. 25

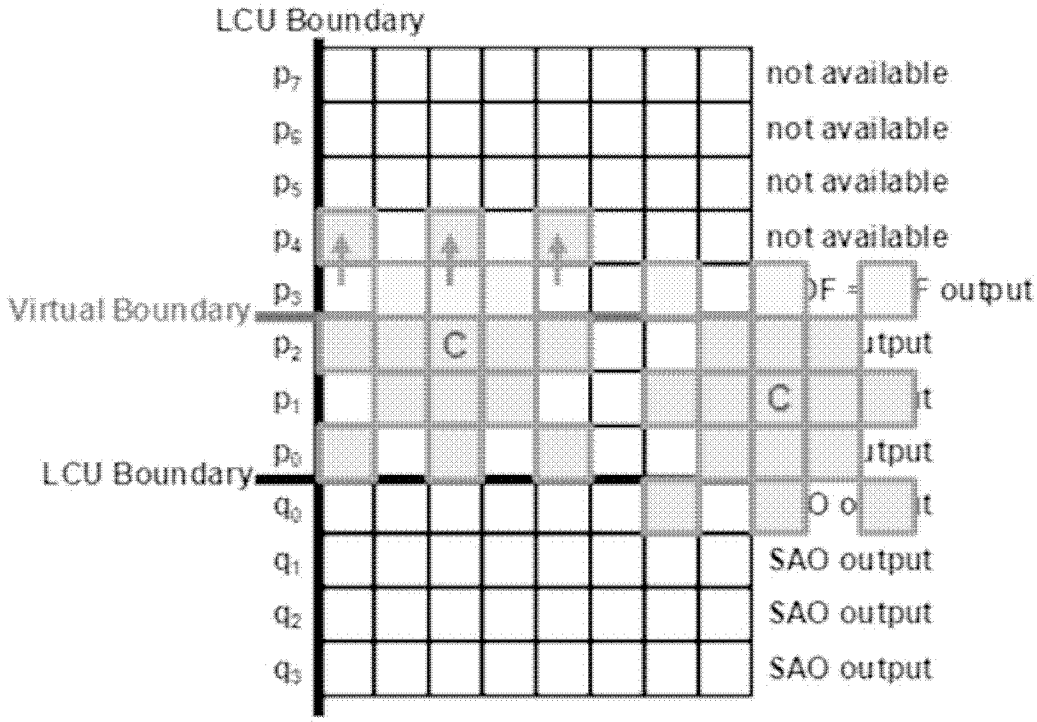
LCU Boundary
p7
p6
p5
p4
p3
Virtual Boundary
p2
p1
p0
LCU Boundary
q0
q1
q2
q3
not available
not available
not available
not available
C
C
SAO output
SAO output
SAO output

Fig. 26

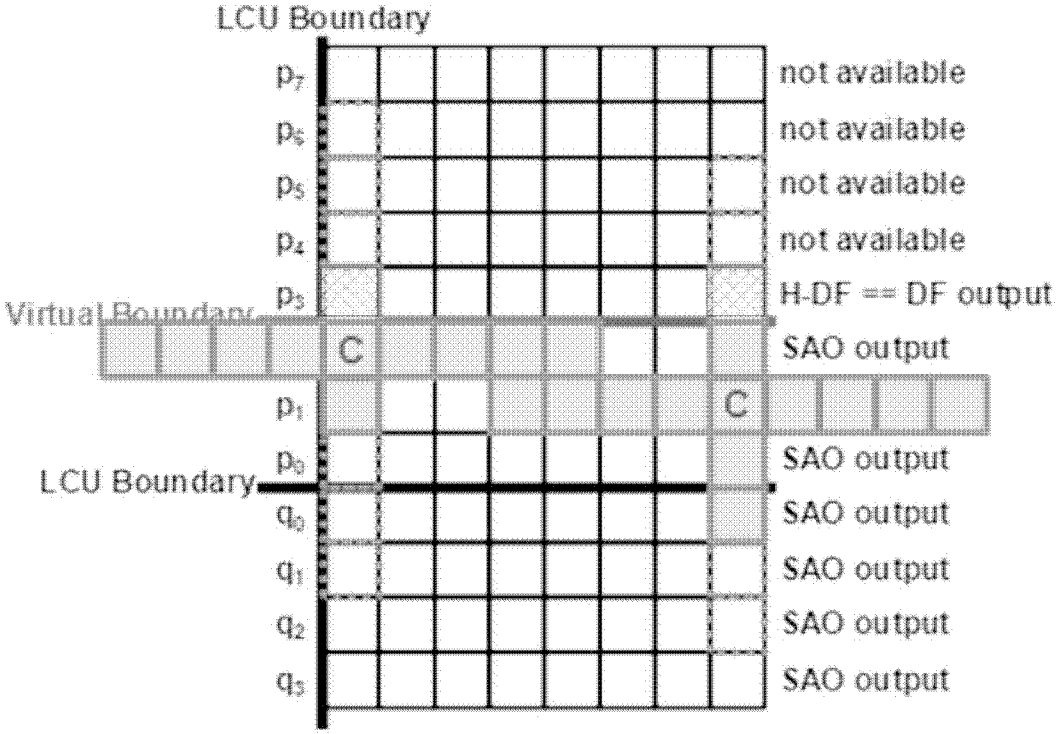
LCU Boundary
p7
p6
p5
p4
p3
Virtual Boundary
p1
p0
LCU Boundary
q0
q1
q2
q3
not available
not available
not available
not available
H-DF == DF output
C
SAO output
C
SAO output
SAO output
SAO output
SAO output
SAO output

Fig. 27

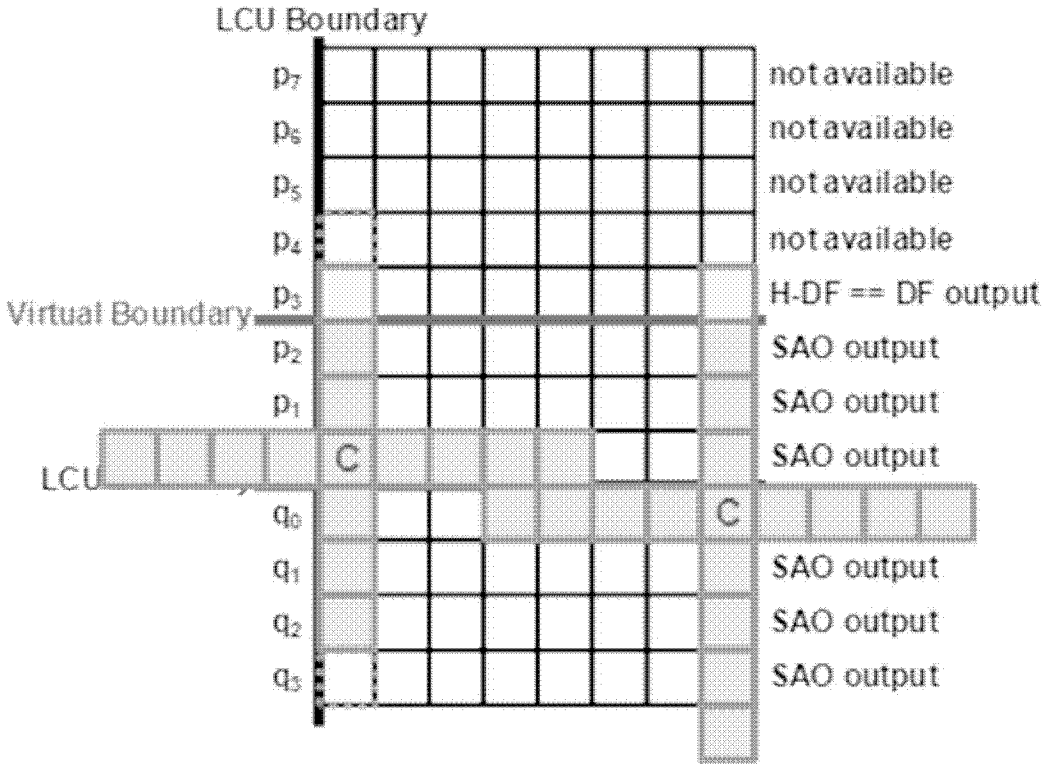
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
p0
q0
q1
q2
q3
not available
not available
not available
not available
H-DF == DF output
SAO output
SAO output
SAO output
SAO output
SAO output
SAO output
Virtual Boundary
LCU
C
C

Fig. 28

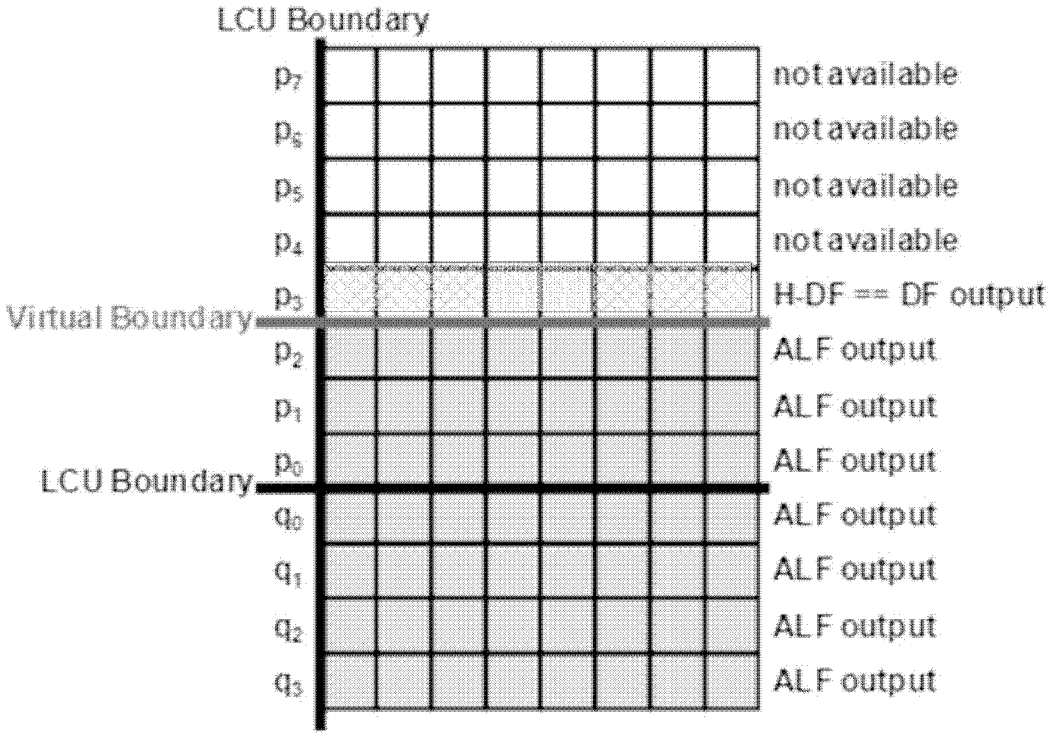
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
p0
q0
q1
q2
q3
not available
not available
not available
not available
H-DF == DF output
ALF output
ALF output
ALF output
ALF output
ALF output
ALF output
ALF output
Virtual Boundary
LCU Boundary

Fig. 29

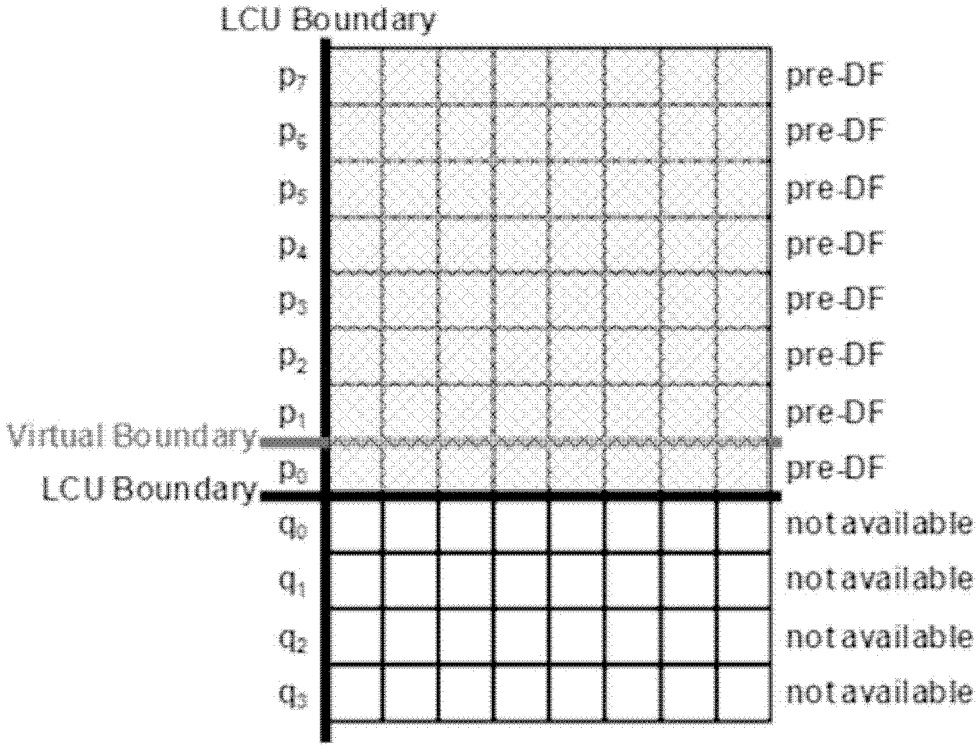
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
Virtual Boundary
p0
LCU Boundary
q0
q1
q2
q3
pre-DF
pre-DF
pre-DF
pre-DF
pre-DF
pre-DF
pre-DF
pre-DF
not available
not available
not available
not available

Fig. 30

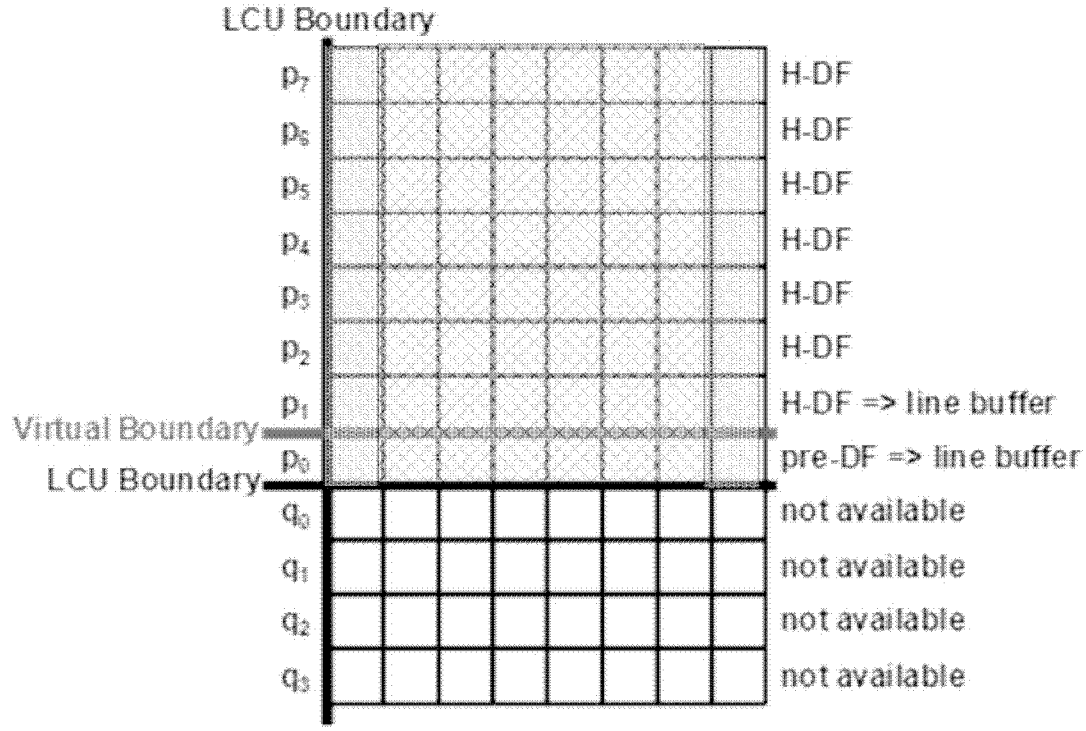
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
Virtual Boundary
p0
LCU Boundary
q0
q1
q2
q3
H-DF
H-DF
H-DF
H-DF
H-DF
H-DF
H-DF => line buffer
pre-DF => line buffer
not available
not available
not available
not available

Fig. 31

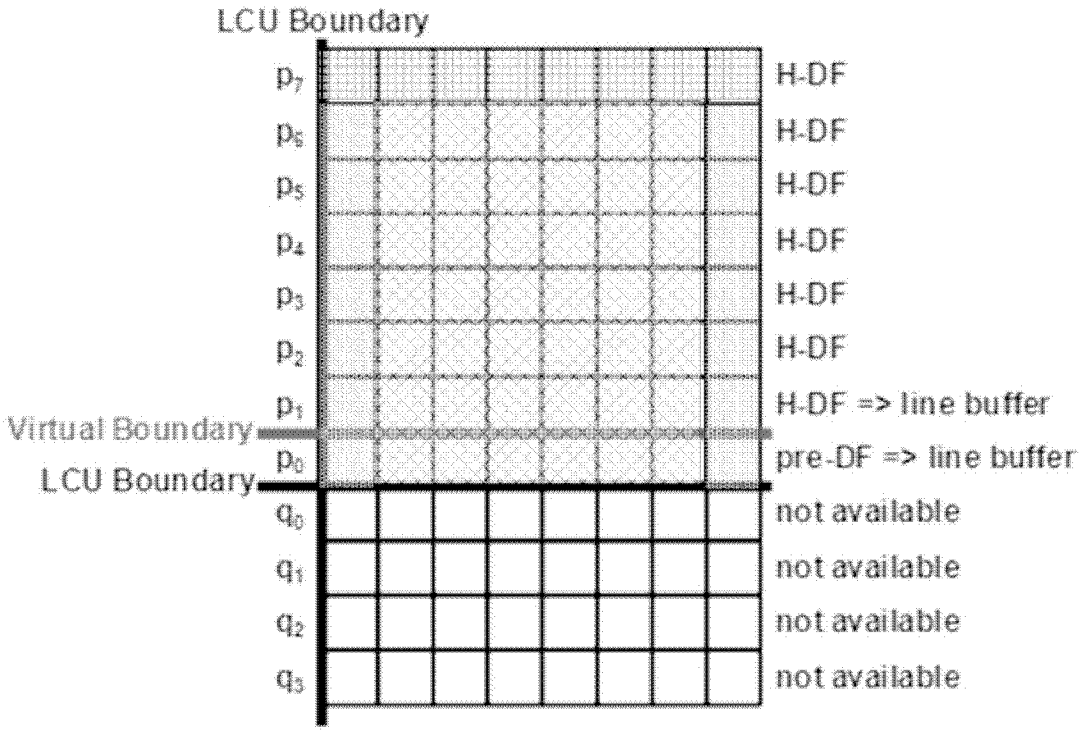
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
Virtual Boundary
p0
LCU Boundary
q0
q1
q2
q3
H-DF
H-DF
H-DF
H-DF
H-DF
H-DF
H-DF => line buffer
pre-DF => line buffer
not available
not available
not available
not available

Fig. 32

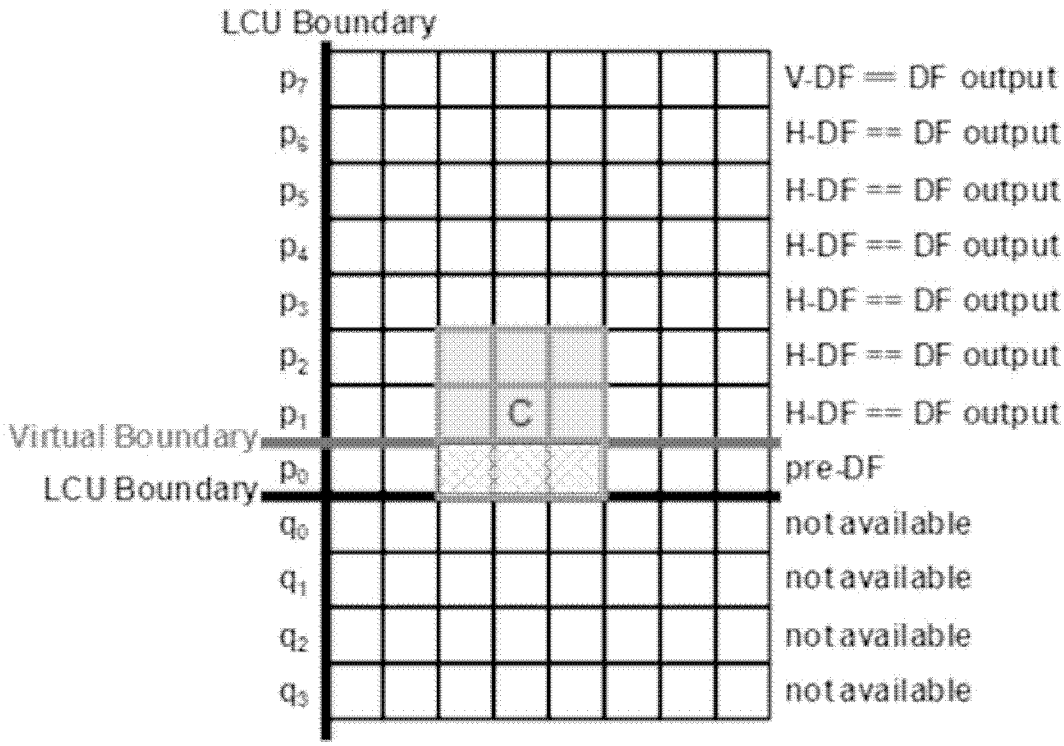
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
Virtual Boundary
p0
LCU Boundary
q0
q1
q2
q3
C
V-DF == DF output
H-DF == DF output
H-DF == DF output
H-DF == DF output
H-DF == DF output
H-DF == DF output
H-DF == DF output
pre-DF
not available
not available
not available
not available

Fig. 33

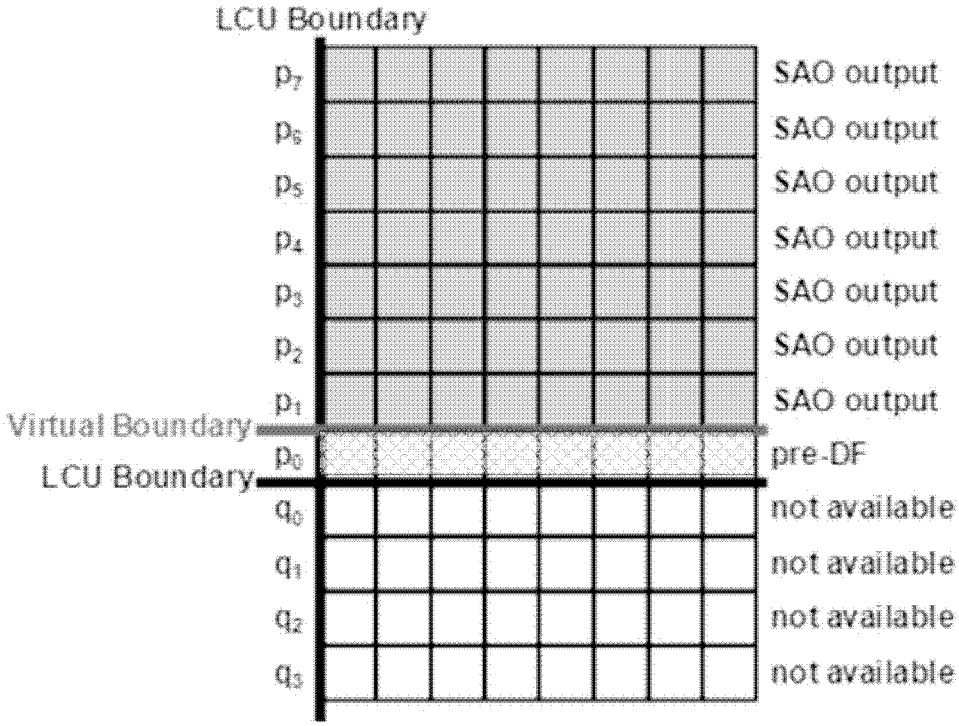
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
Virtual Boundary
p0
LCU Boundary
q0
q1
q2
q3
SAO output
SAO output
SAO output
SAO output
SAO output
SAO output
SAO output
pre-DF
not available
not available
not available
not available

Fig. 34

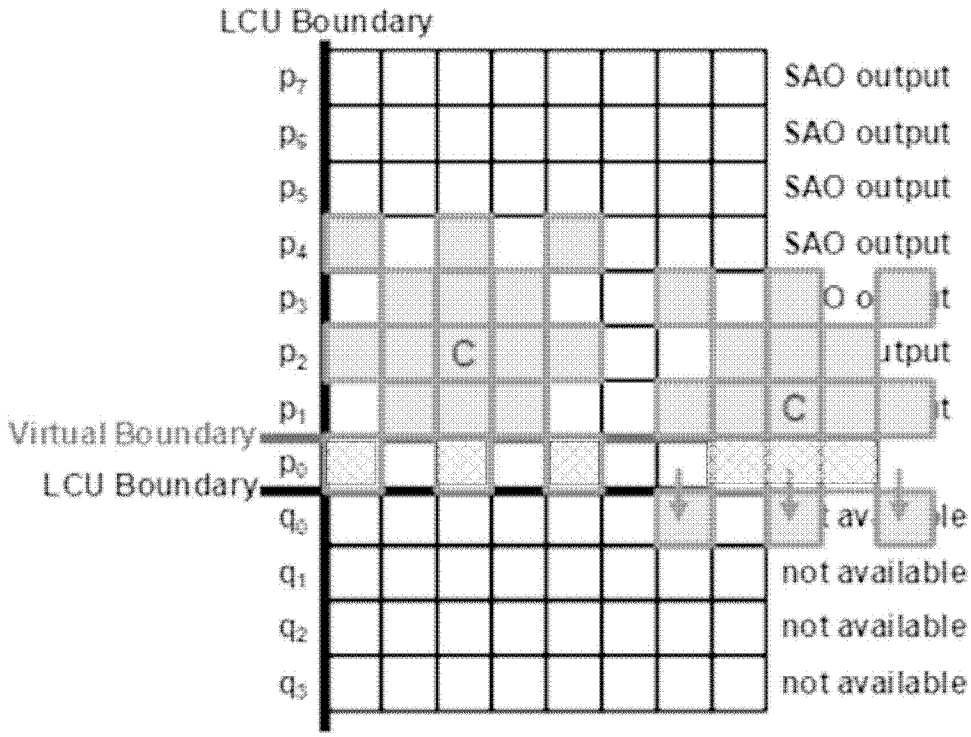
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
Virtual Boundary
p0
LCU Boundary
q0
q1
q2
q3
SAO output
SAO output
SAO output
SAO output
not available
not available
not available

Fig. 35

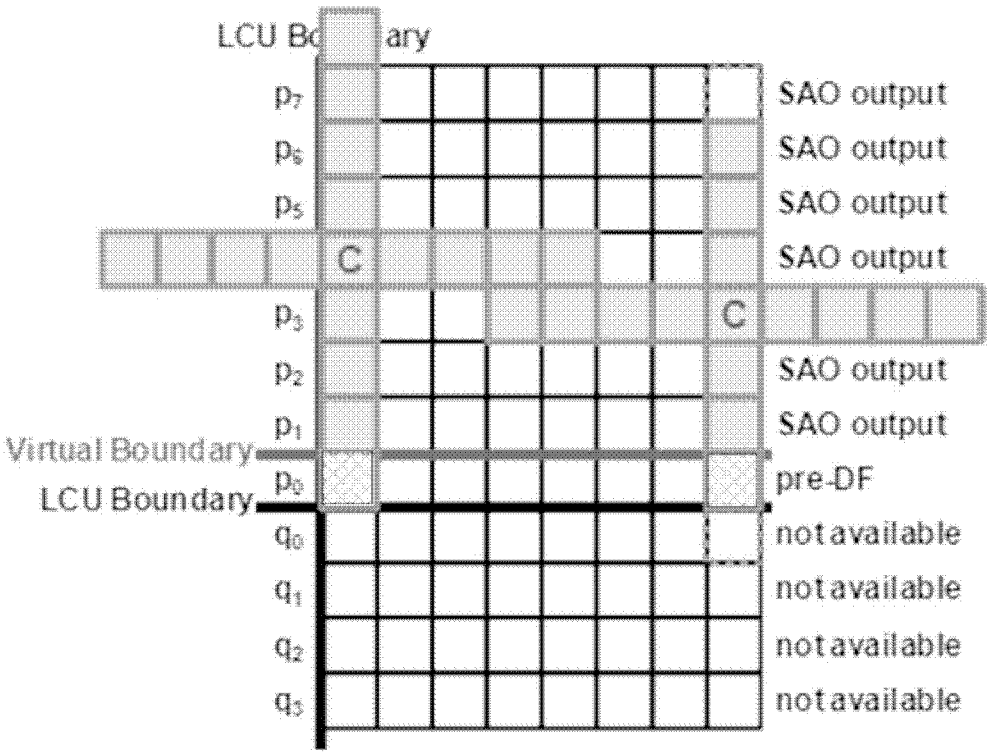
LCU Boundary
p7
p6
p5
C
p3
p2
p1
Virtual Boundary
p0
LCU Boundary
q0
q1
q2
q3
C
SAO output
SAO output
SAO output
SAO output
SAO output
SAO output
pre-DF
not available
not available
not available
not available

Fig. 36

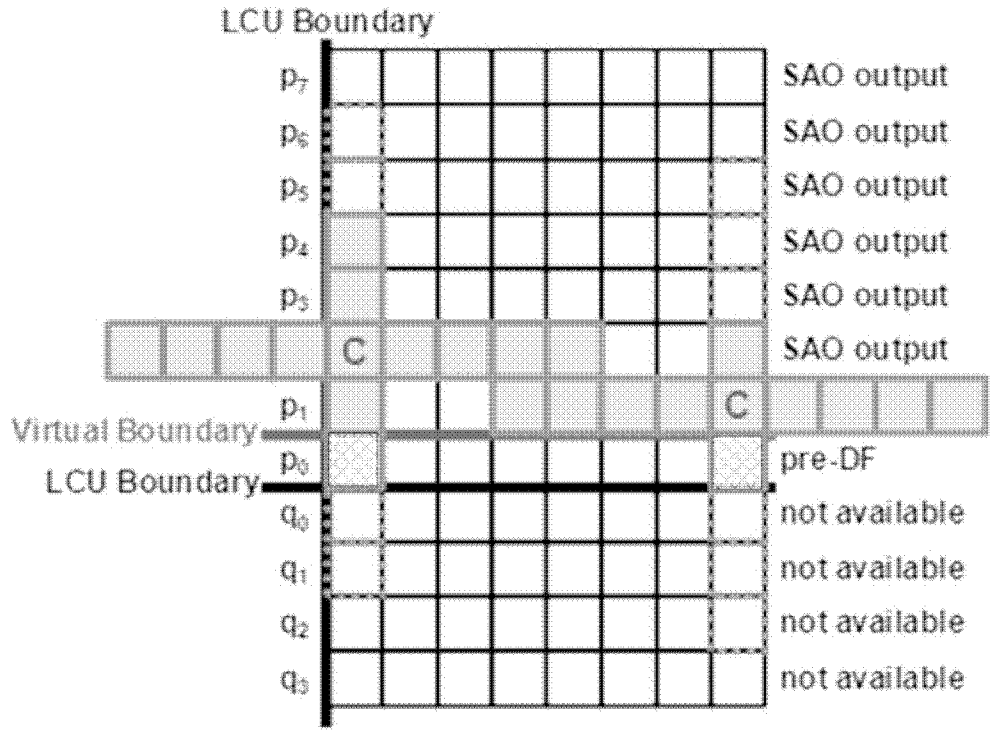
LCU Boundary
p7
p6
p5
p4
p3
C
p1
Virtual Boundary
p0
LCU Boundary
q0
q1
q2
q3
C
SAO output
SAO output
SAO output
SAO output
SAO output
SAO output
pre-DF
not available
not available
not available
not available

Fig. 37

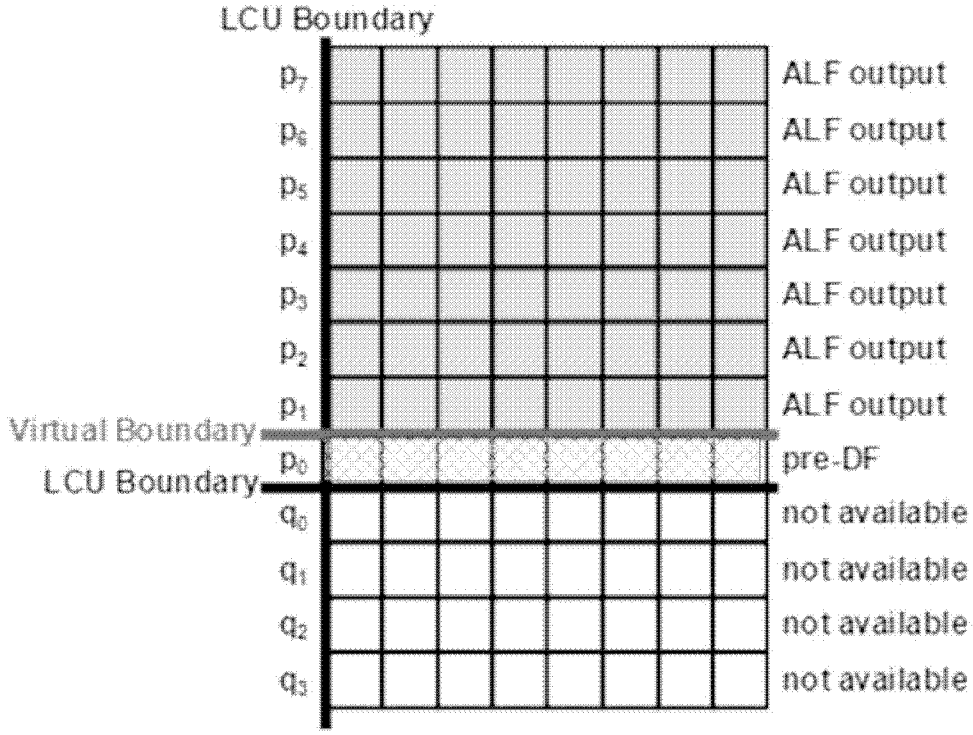
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
Virtual Boundary
p0
LCU Boundary
q0
q1
q2
q3
ALF output
ALF output
ALF output
ALF output
ALF output
ALF output
ALF output
pre-DF
not available
not available
not available
not available

Fig. 38

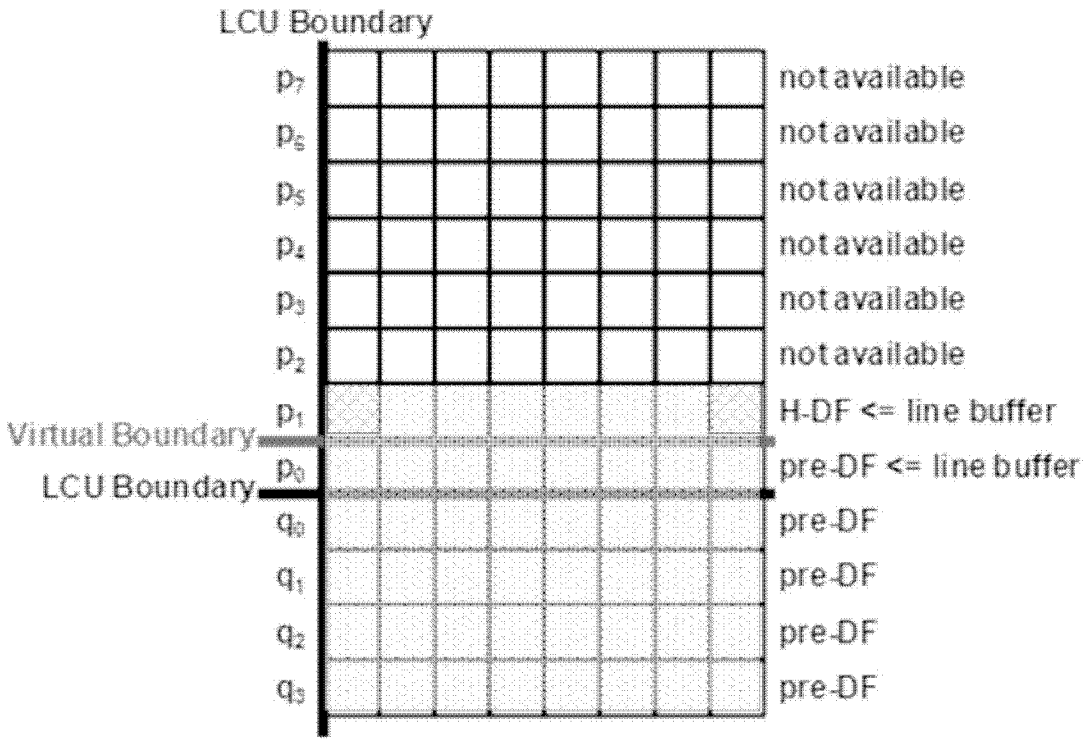
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
Virtual Boundary
p0
LCU Boundary
q0
q1
q2
q3
not available
not available
not available
not available
not available
not available
H-DF <= line buffer
pre-DF <= line buffer
pre-DF
pre-DF
pre-DF
pre-DF

Fig. 39

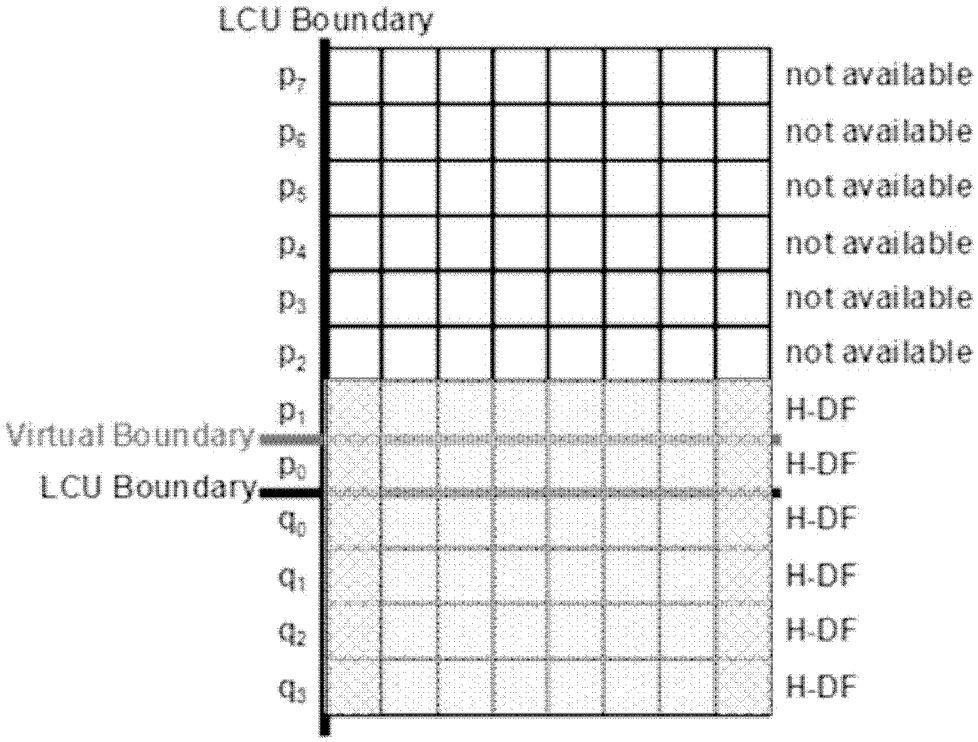
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
p0
q0
q1
q2
q3
not available
not available
not available
not available
not available
not available
H-DF
H-DF
H-DF
H-DF
H-DF
H-DF
Virtual Boundary
LCU Boundary

Fig. 40

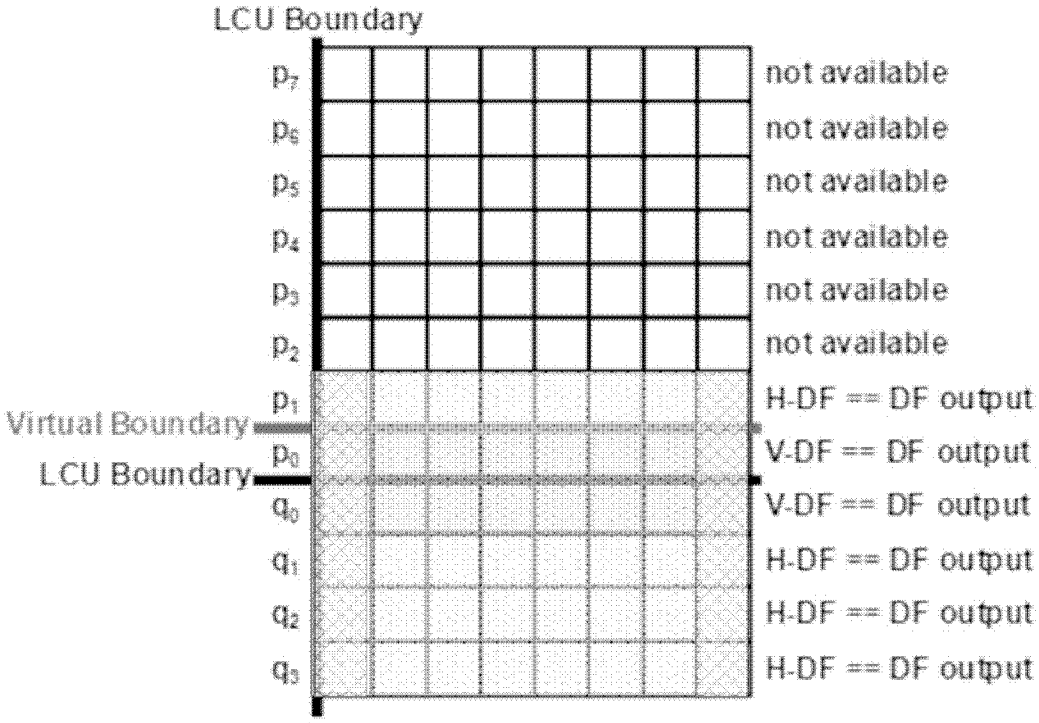
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
p0
q0
q1
q2
q3
not available
not available
not available
not available
not available
not available
H-DF == DF output
V-DF == DF output
V-DF == DF output
H-DF == DF output
H-DF == DF output
H-DF == DF output
Virtual Boundary
LCU Boundary

Fig. 41

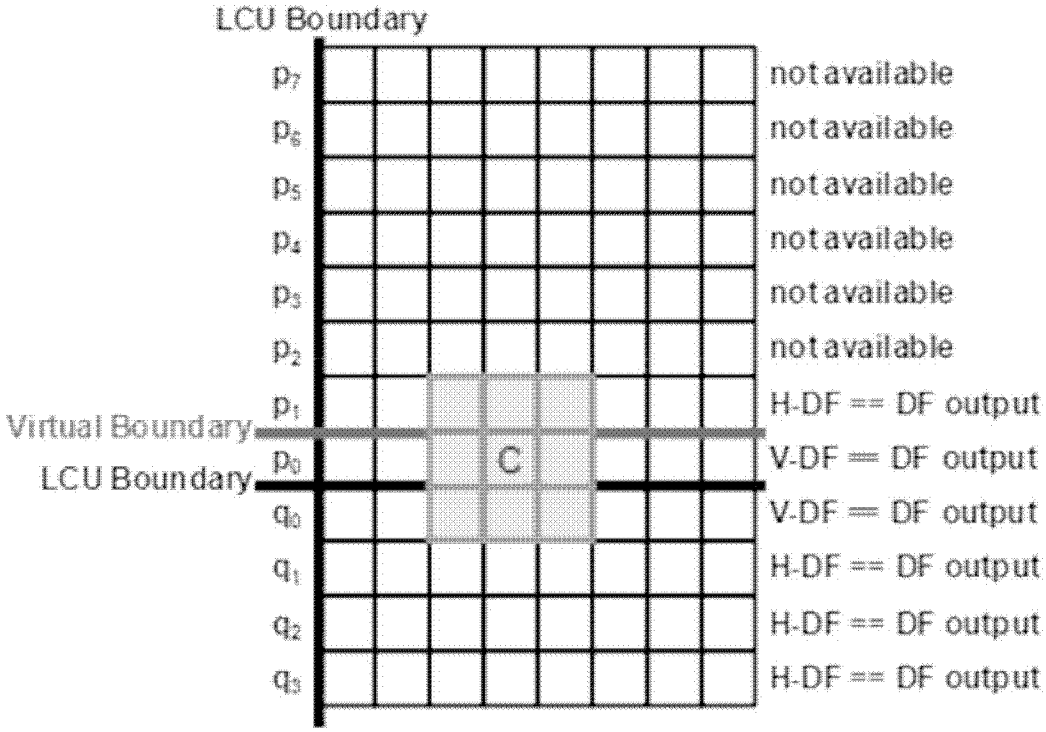
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
Virtual Boundary
p0
LCU Boundary
q0
q1
q2
q3
C
not available
not available
not available
not available
not available
not available
H-DF == DF output
V-DF == DF output
V-DF == DF output
H-DF == DF output
H-DF == DF output
H-DF == DF output

Fig. 42

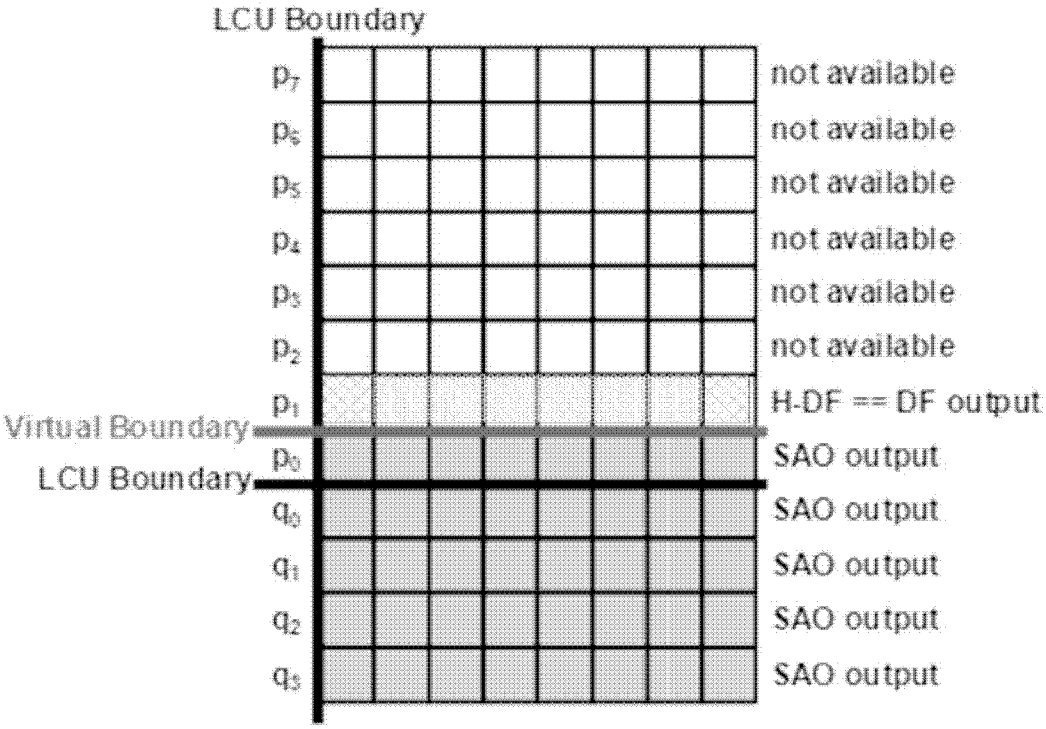
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
Virtual Boundary
p0
LCU Boundary
q0
q1
q2
q3
not available
not available
not available
not available
not available
not available
H-DF == DF output
SAO output
SAO output
SAO output
SAO output
SAO output

Fig. 43

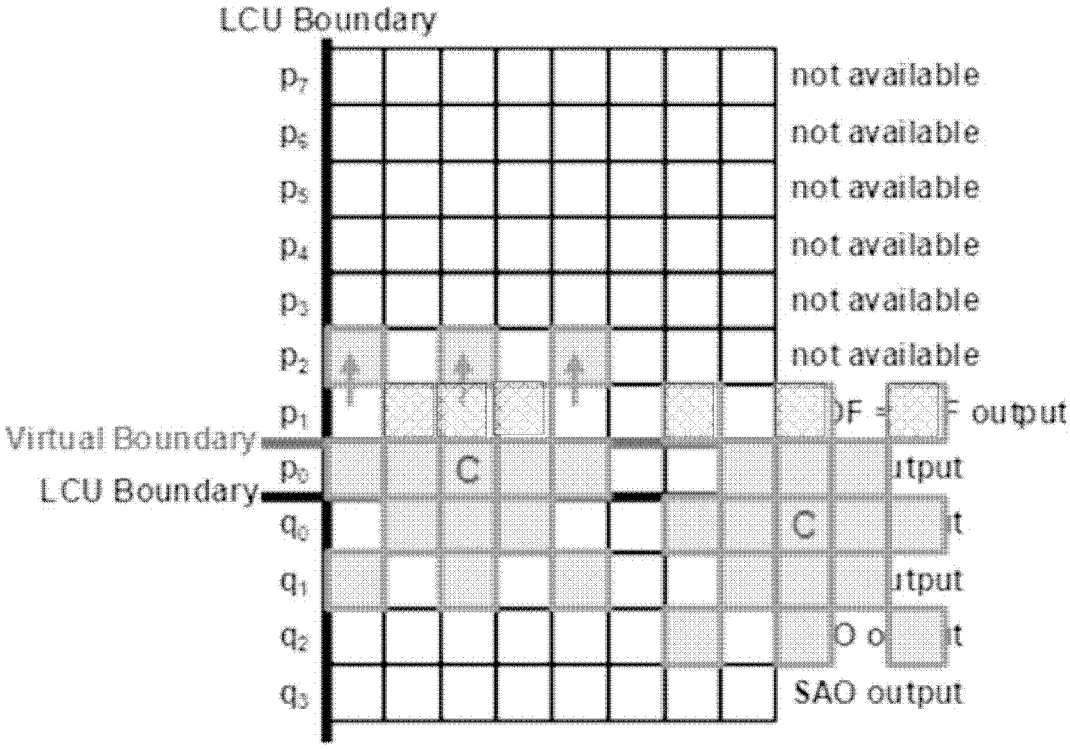
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
p0
q0
q1
q2
q3
not available
not available
not available
not available
not available
not available
Virtual Boundary
LCU Boundary
C
C
SAO output

Fig. 44

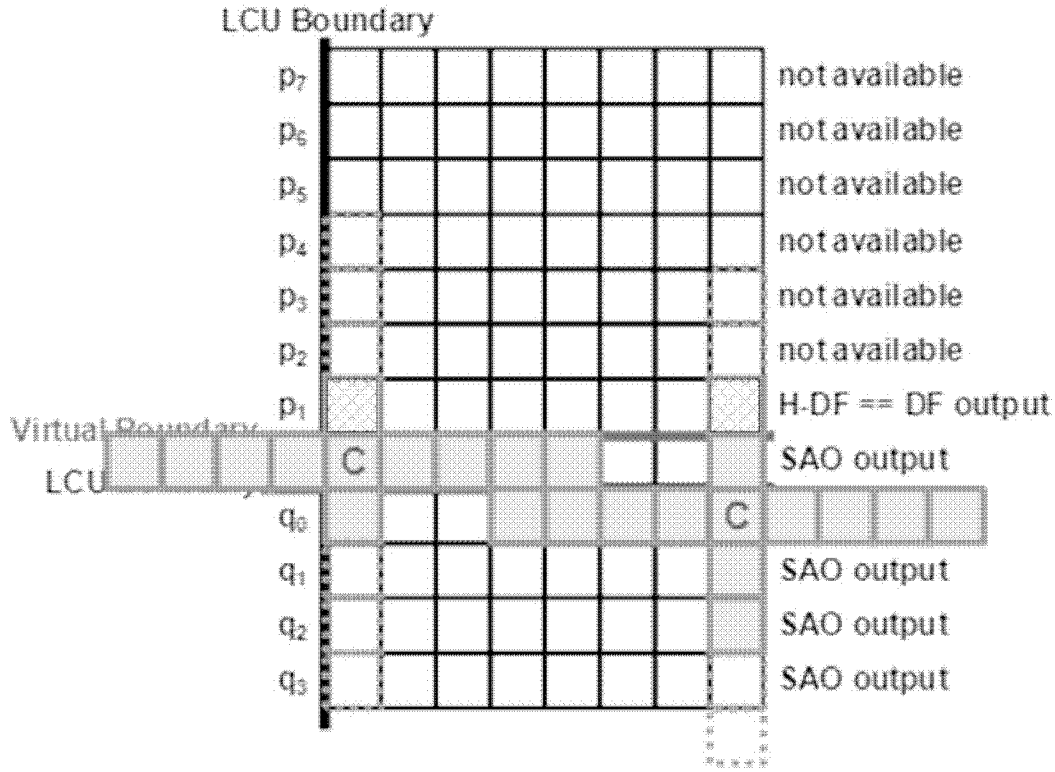
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
p0
q0
q1
q2
q3
not available
not available
not available
not available
not available
not available
H-DF == DF output
SAO output
C
C
SAO output
SAO output
SAO output

Fig. 45

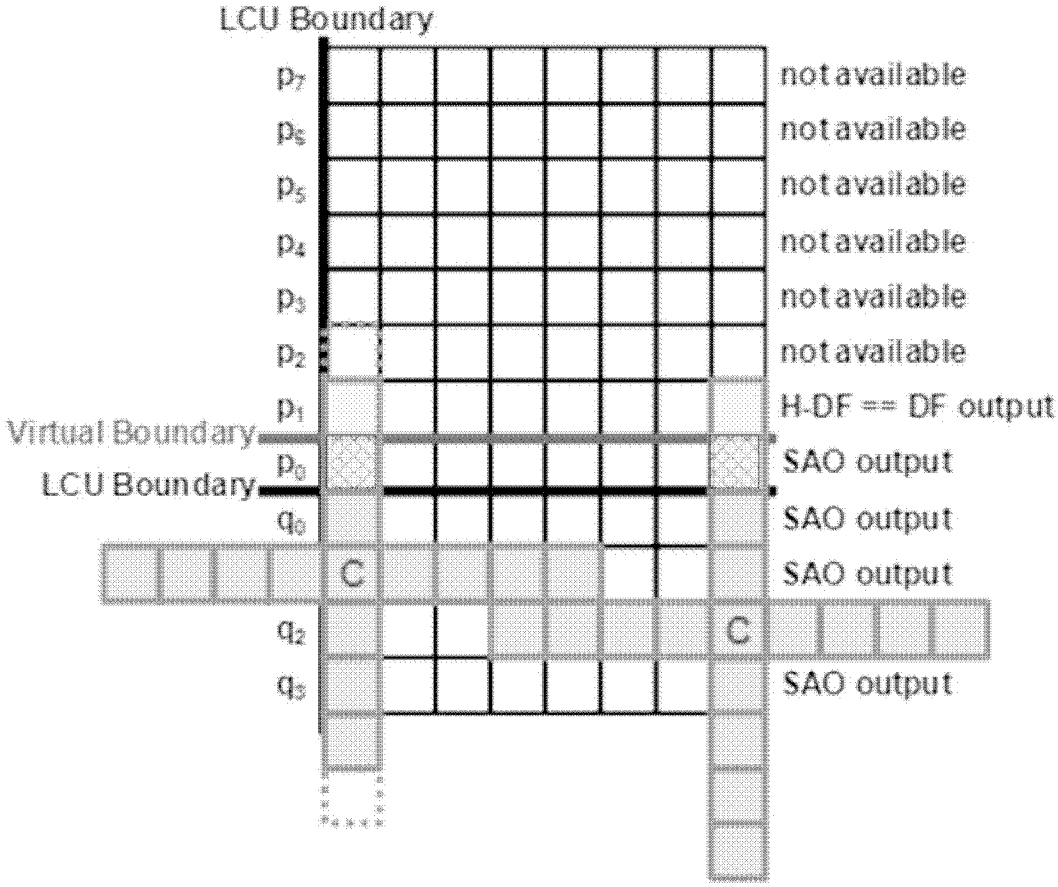
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
Virtual Boundary
p0
LCU Boundary
q0
q2
q3
C
C
not available
not available
not available
not available
not available
not available
H-DF == DF output
SAO output
SAO output
SAO output
SAO output

Fig. 46

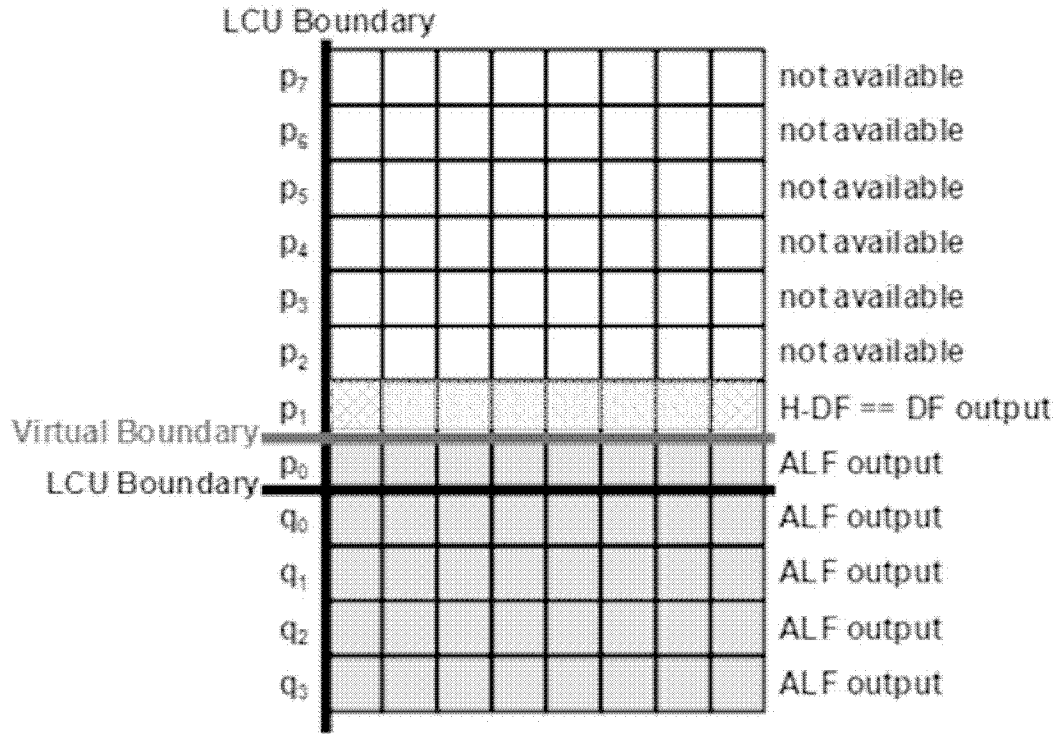
LCU Boundary
p7
p6
p5
p4
p3
p2
p1
Virtual Boundary
p0
LCU Boundary
q0
q1
q2
q3
not available
not available
not available
not available
not available
not available
H-DF == DF output
ALF output
ALF output
ALF output
ALF output
ALF output

Fig. 47

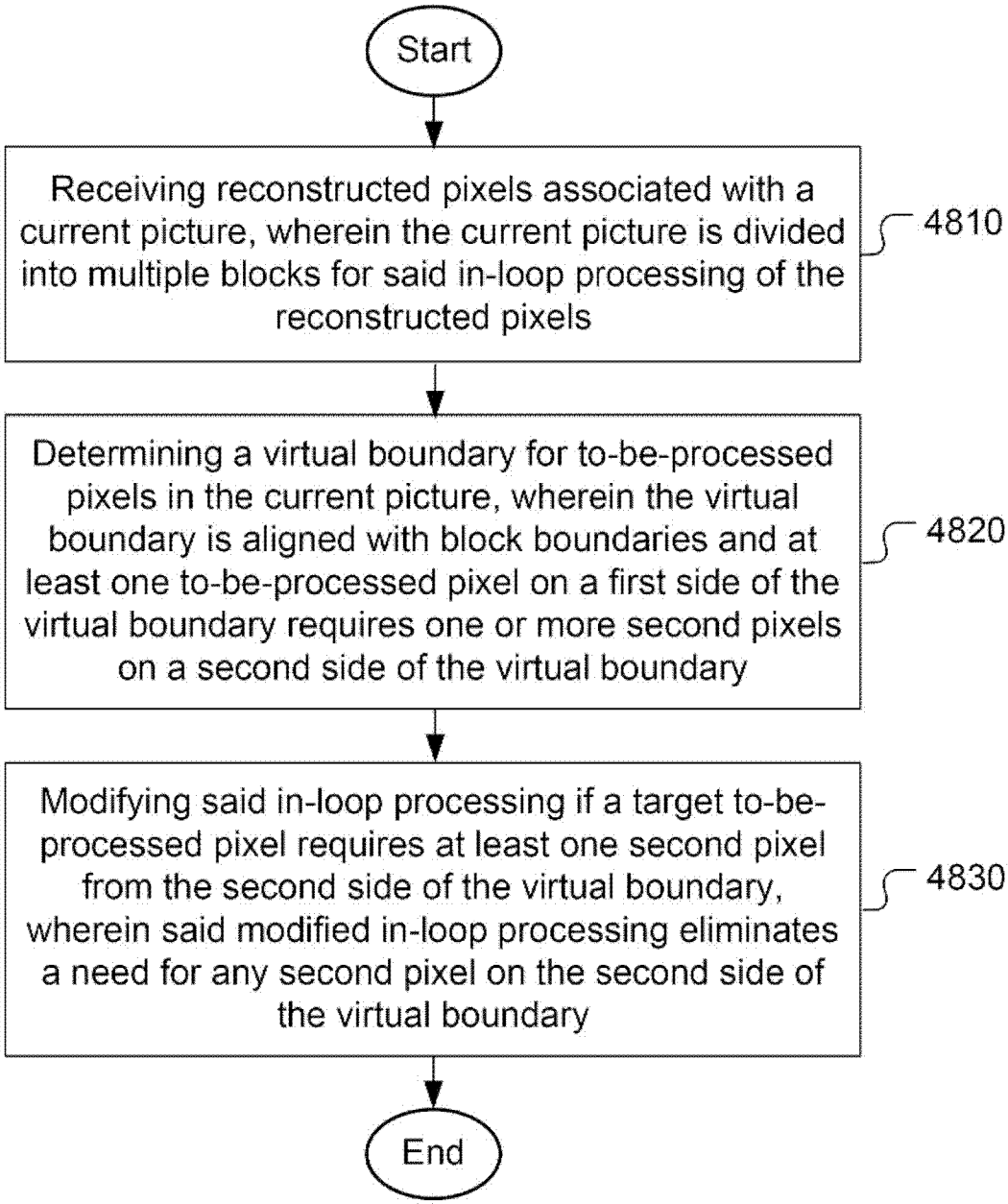
Start
Receiving reconstructed pixels associated with a current picture, wherein the current picture is divided into multiple blocks for said in-loop processing of the reconstructed pixels
4810
Determining a virtual boundary for to-be-processed pixels in the current picture, wherein the virtual boundary is aligned with block boundaries and at least one to-be-processed pixel on a first side of the virtual boundary requires one or more second pixels on a second side of the virtual boundary
4820
Modifying said in-loop processing if a target to-be-processed pixel requires at least one second pixel from the second side of the virtual boundary, wherein said modified in-loop processing eliminates a need for any second pixel on the second side of the virtual boundary
4830
End

Fig. 48

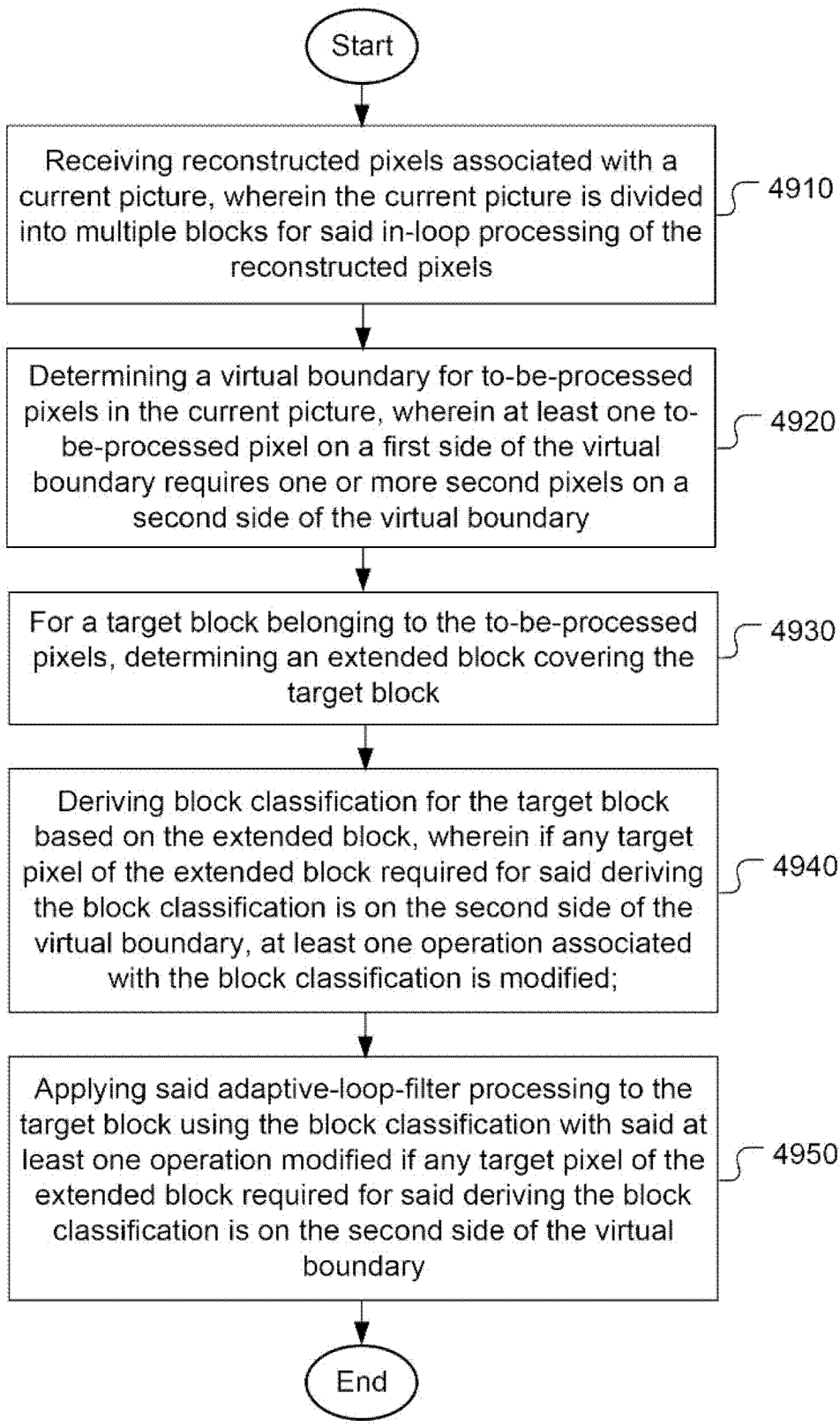
Start
Receiving reconstructed pixels associated with a current picture, wherein the current picture is divided into multiple blocks for said in-loop processing of the reconstructed pixels
4910
Determining a virtual boundary for to-be-processed pixels in the current picture, wherein at least one to-be-processed pixel on a first side of the virtual boundary requires one or more second pixels on a second side of the virtual boundary
4920
For a target block belonging to the to-be-processed pixels, determining an extended block covering the target block
4930
Deriving block classification for the target block based on the extended block, wherein if any target pixel of the extended block required for said deriving the block classification is on the second side of the virtual boundary, at least one operation associated with the block classification is modified;
4940
Applying said adaptive-loop-filter processing to the target block using the block classification with said at least one operation modified if any target pixel of the extended block required for said deriving the block classification is on the second side of the virtual boundary
4950
End

Fig. 49

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2019/111895

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/70(2014.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

VEN;CNABS;CNTXT;USTXT;EPTXT;WOTXT;CNKI:virtual, boundary, adaptive, loop, in-loop, buffer, classif+, filter, side, reconstruct, block, modify, target, pixel, eliminate

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CN 103503456 A (MEDIATEK INC) 08 January 2014 (2014-01-08) description, paragraphs 4-43 and figures 1A-24	1-12
A	CN 105794206 A (MEDIATEK INC) 20 July 2016 (2016-07-20) the whole document	1-23
A	CN 103891292 A (MEDIATEK INC) 25 June 2014 (2014-06-25) the whole document	1-23
A	CN 106028050 A (MEDIATEK INC) 12 October 2016 (2016-10-12) the whole document	1-23
A	US 2012082244 A1 (CHEN CHING-YEH et. al.) 05 April 2012 (2012-04-05) the whole document	1-23

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
18 December 2019	**08 January 2020**
Name and mailing address of the ISA/CN **National Intellectual Property Administration, PRC** **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088** **China** Facsimile No. **(86-10)62019451**	Authorized officer **WANG,Qian** Telephone No. **86-（010）-62412164**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2019/111895

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
CN	103503456	A	08 January 2014	CN	103503456	B	22 March 2017
				EP	2708027	A1	19 March 2014
				WO	2012152174	A1	15 November 2012
				US	2013322523	A1	05 December 2013
				KR	20140029436	A	10 March 2014
				EP	2708027	A4	22 April 2015
				JP	2014517555	A	17 July 2014
				KR	101567467	B1	09 November 2015
				JP	5689563	B2	25 March 2015
CN	105794206	A	20 July 2016	EP	3069511	A1	21 September 2016
				EP	3069512	A1	21 September 2016
				WO	2015070772	A1	21 May 2015
				EP	3069511	A4	27 September 2017
				US	10123048	B2	06 November 2018
				WO	2015070739	A1	21 May 2015
				EP	3069512	A4	14 June 2017
				US	2016261863	A1	08 September 2016
				US	2016269753	A1	15 September 2016
				CN	105850121	A	10 August 2016
				CN	105794206	B	28 December 2018
				CA	2929894	A1	21 May 2015
				CN	105850121	B	12 October 2018
CN	103891292	A	25 June 2014	EP	2737705	A1	04 June 2014
				CN	103891292	B	02 February 2018
				AU	2012327672	B2	03 September 2015
				WO	2013060250	A1	02 May 2013
				AU	2012327672	A1	20 March 2014
				US	2014198844	A1	17 July 2014
				EP	2737705	A4	02 March 2016
CN	106028050	A	12 October 2016	CN	106028050	B	26 April 2019
				GB	2500347	B	16 May 2018
				CN	103535035	B	15 March 2017
				CN	103535035	A	22 January 2014
				GB	2500347	A	18 September 2013
				GB	201311592	D0	14 August 2013
				ZA	201305528	B	29 October 2014
				WO	2012155553	A1	22 November 2012
				CN	105120270	B	04 September 2018
				DE	112012002125	T5	20 February 2014
				CN	105120270	A	02 December 2015
US	2012082244	A1	05 April 2012	US	8861617	B2	14 October 2014