



(51) International Patent Classification:

G06K 9/36 (2006.01) H04N 7/36 (2006.01)
H04N 7/12 (2006.01) H04N 11/02 (2006.01)

(21) International Application Number:

PCT/US2016/039094

(22) International Filing Date:

23 June 2016 (23.06.2016)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

14/750,762 25 June 2015 (25.06.2015) US

(71) Applicant: SONY INTERACTIVE ENTERTAINMENT INC. [US/JP]; 1-7-1 Konan, Minato-ku, Tokyo, Tokyo 108-0075 (JP).

(72) Inventors: KRISHNAN, Rathish; c/o Sony Interactive Entertainment Inc., 2207 Bridgepointe Pkwy, San Mateo, California 94404 (US). LEE, Hung-Ju; c/o Sony Interact-

ive Entertainment Inc., 2207 Bridgepointe Pkwy, San Mateo, California 94404 (US).

(74) Agent: ISENBERG, Joshua D.; c/o JDI Patent, 809 Corporate Way, Fremont, California 94539 (US).

(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, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, 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,

[Continued on next page]

(54) Title: PARALLEL INTRA-PREDICTION

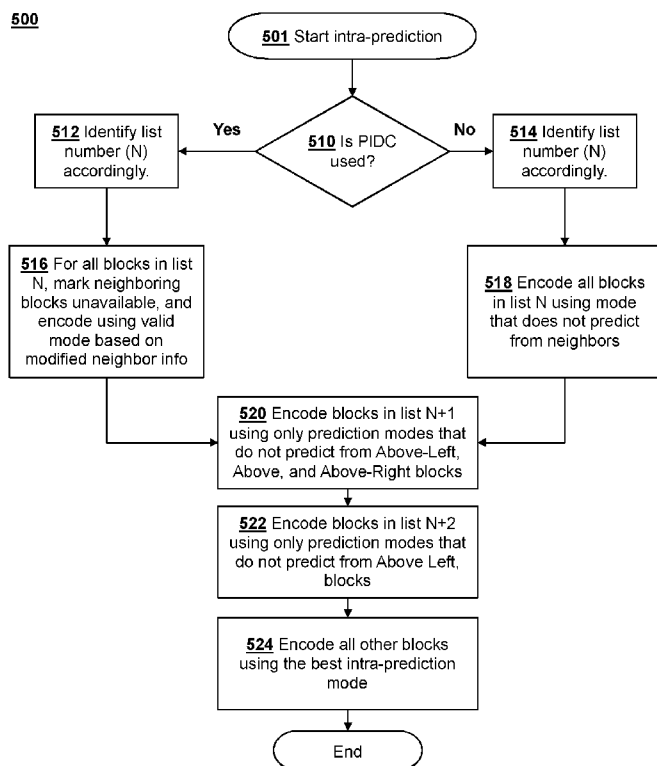


FIG. 5

(57) Abstract: Digital pictures may be encoded and decoded in a way that optimizes parallelization. Selected sections of a digital picture in a first list are encoded independently of encoding any of their neighbor sections. Sections in a second list located to the right of corresponding sections in the first list are encoded using at most the section to their immediate left. Sections in a third list located to the right of a corresponding section in the second list and immediately below a corresponding section in the first list are encoded using at most the sections to the immediate left, immediately above and immediately to its above-right. All other sections of the picture may be encoded in a normal manner.



DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, **Published:**

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).

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

PARALLEL INTRA-PREDICTION

FIELD OF THE DISCLOSURE

This application claims the priority benefit of commonly-assigned U.S. patent application number 14/750,762 filed June 25, 2015, the entire disclosures of which are incorporated
5 herein by reference.

Aspects of the present disclosure are related to encoding and decoding compressed bitstreams. In particular, aspects of the present disclosure are related to strategies for the creation of bitstreams that allow parallel intra-prediction for encoding and decoding.

BACKGROUND

10 Video encoding and decoding involves several processes that can be parallelized and be effectively implemented on multi-processors and a graphics processing unit (GPU). Motion estimation and Transform/Scaling are examples of processes that have no dependencies on neighboring blocks, making it highly suitable for implementing on the GPU. Other processes such as intra-prediction and deblocking, however, have dependencies on neighboring blocks.

15 For H.264 intra-prediction, there are nine luma prediction modes for the 4x4 block size and four modes for 16x16 block size. Each 4x4 block has 16 luma or chroma samples and each 16x16 block has 256 luma samples. A luma sample represents the monochrome signal and a chroma sample represents one of the two color difference signals related to the primary colors. For the 4x4 luma intra-prediction, depending on the mode, a macroblock may predict
20 from the following neighbors as shown in FIG. 1A:

- Above-Left Macroblock
- Above Macroblock
- Above-Right Macroblock
- Left Macroblock

25 Having dependencies on neighbors restricts the number of macroblocks that can be processed in parallel. It also necessitates synchronization points that ensure that macroblocks are not processed before their neighbors are ready. The number of macroblocks that are processed between the synchronization steps is not constant, and it takes several synchronization steps before peak parallelism is reached in terms of macroblock processing rate. It is difficult to
30 parallelize intra-prediction for a decoder. In particular, efficient parallelization is not

currently possible if there is only one slice per frame, which may happen if the encoder does not create bitstreams tailored to maximize the performance of the decoder. These issues are not eliminated even when decoding multiple slices with a GPU. Due to these issues, processes such as intra-prediction cannot make full use of a GPU's processing capability.

- 5 It is within this context that aspects of the present disclosure arise.

BRIEF DESCRIPTION OF THE DRAWINGS

The teachings of the present disclosure can be readily understood by considering the following detailed description in conjunction with the accompanying drawings, in which:

- 10 FIG. 1A is a schematic diagram of a portion of a digital picture illustrating an example of data dependencies for 4x4 Intra-prediction.

FIG. 1B is a schematic diagram illustrating one possible division of a video data picture within the context of the present disclosure.

FIG. 2 is a schematic diagram illustrating conventional single-slice wavefront block processing order.

- 15 FIG. 3 is a graph illustrating the number of blocks that can be processed conventionally during each run of a compute shader that performs Intra-prediction for a 1080p picture with worst-case dependency, where the basic coding unit is assumed to be 16x16.

- 20 FIG. 4A is a schematic diagram of a portion of a digital picture illustrating an encoding strategy used to create bitstreams that allow parallel intra-prediction in accordance with aspects of the present disclosure.

FIG. 4B is a schematic diagram of a portion of a digital picture illustrating an alternative encoding strategy used to create bitstreams that allow parallel intra-prediction in accordance with aspects of the present disclosure.

- 25 FIG. 5 is a flow diagram illustrating an example of picture encoding in accordance with aspects of the present disclosure.

FIG. 6 is a flow diagram illustrating an example of decoding a picture encoded in accordance with aspects of the present disclosure.

FIG. 7 is a graph illustrating the number of blocks that can be processed during each run of a compute shader that performs Intra-prediction in accordance with aspects of the present disclosure for a hypothetical example of a 1080p picture with worst-case dependency.

FIG. 8 is a schematic diagram illustrating the mode selection process when PIPCM mode is used to encode a digital picture in accordance with aspects of the present disclosure.

FIG. 9 is a schematic diagram illustrating the mode selection process when PIDC mode is used to encode a digital picture in accordance with aspects of the present disclosure.

FIG. 10 is a schematic diagram of a digital picture divided into a conventional multiple-slice configuration with uniform number of blocks per slice in which the blocks within a slice can be encoded or decoded independent of blocks within other slices.

FIG. 11 is a schematic diagram of a digital picture divided into an alternative multiple-slice configuration in which multiple groups of blocks within each slice can be independently encoded or decoded in accordance with aspects of the present disclosure.

FIG. 12 is a schematic diagram of a digital picture divided into a single-slice configuration in which multiple groups of blocks within the slice can be encoded or decoded independent of one another in accordance with aspects of the present disclosure.

FIG. 13 is a block diagram of a device having an encoder and decoder capable of parallel intra-prediction encoding and decoding in accordance with aspects of the present disclosure

DESCRIPTION OF THE DRAWINGS

Although the following detailed description contains many specific details for the purposes of illustration, anyone of ordinary skill in the art will appreciate that many variations and alterations to the following details are within the scope of the invention. Accordingly, the exemplary embodiments of the invention described below are set forth without any loss of generality to, and without imposing limitations upon, the claimed invention.

INTRODUCTION

In the context of embodiments of the present invention, video data may be broken down in suitable sized units for coding and decoding. For example, in the case of video data, the video data may be broken down into pictures with each picture representing a particular image in a series of images. Each unit of video data may be broken down into sub-units of

varying size. Generally, within each unit there is some smallest or fundamental sub-unit. In the case of video data, each video frame may be broken down into pixels, each of which contains luma (brightness) and chroma (color) data.

5 By way of example, and not by way of limitation, as shown in FIG. 1B, a single picture **100** (e.g., a digital video frame) may be broken down into one or more sections. As used herein, the term “section” can refer to a group of one or more luma or chroma samples within the picture **100**. A section can range from a single luma or chroma sample within the picture, up to the whole picture. Non-limiting examples of sections include, but are not limited to slices
10 **102**, blocks **104** (e.g., 16X16 sample), sub-blocks **106** (e.g., 8X8 sample), partitions **108** (e.g., 4X4 sample) and individual luma or chroma samples **110**. As illustrated in FIG. 1B, each slice **102** contains one or more rows of blocks **104** or portions of one or more such rows. The number of blocks in a row depends on the size of the blocks and the size and resolution of the picture **100**. For example, if each block contains sixteen by sixteen luma samples then the
15 number of blocks in each row may be determined by dividing the width of the picture **100** (in luma samples) by sixteen. Each block **104** may be broken down into a number of sub-blocks **106**. Each sub-block **106** may be broken down into a number of partitions **108** and each partition may contain a number of luma or chroma samples **110**. By way of example, and without limitation of the invention, in a common video coding scheme, each block **104** may
20 be broken down into four sub-blocks **106**. Each sub-block may be broken down into four partitions **108** and each partition may contain a four by four arrangement of sixteen luma or chroma samples **110**.

Conventionally, video images have been decoded using a single thread for the decoding of
25 each picture or the decoding of each slice. In a conventional single-thread decoder all decoding tasks for one block are completed before decoding the next block. Multithreaded picture decoding – i.e., decoding multiple pictures in parallel, may also be done. To facilitate multi-threaded decoding within pictures, each picture **100** may be broken down into two or more sections **112** that encompass one or more slices.

30

The video decoding of H.264/AVC or HEVC compressed bitstreams requires significant computing power. For high resolution or high quality video bitstreams, pure CPU based solutions might not achieve real-time decoding performance. Performing some video

decoding processes on the GPU, while using the CPU to perform other video decoding tasks, is a popular approach that can speed up the video decoding process.

GPUs are extremely efficient at computations which have high arithmetic intensity, involve large data sets and have few data dependencies. While the first two constraints are satisfied for most video decoding tasks, the last constraint is not. Decoding processes such as intra-prediction and deblocking involve dependencies on neighboring blocks. For intra-prediction, a block may have dependencies on one or more of four adjacent blocks. Due to these dependencies, implementing intra-prediction on the GPU is inefficient and for worst-case dependency, there are periods where only a small number blocks are actively processed on the GPU. Aspects of the present disclosure improve decoder performance for worst-case dependency and for typical cases as well.

Algorithms used in image and video processing that have dependencies on neighboring blocks are difficult to implement efficiently on multi-processors, which require large data sets that can be processed independently with minimal or no synchronization. Intra-prediction, as defined in the H.264 or HEVC standard, is one example of a process that is challenging to implement on the GPU due to the dependencies of a block on neighbors. Existing techniques make such modules GPU-friendly by processing blocks using a processing order as shown in FIG. 2. A block numbered N can be processed once neighboring blocks smaller than N are processed. All blocks represented with the same number can be processed in parallel. For simplicity, the blocks that can be processed in parallel are represented by a list corresponding to the block number. From FIG. 2, List 8 would consist of four blocks, and List 15 would consist of eight blocks. Each list includes all blocks that can be encoded simultaneously and decoded simultaneously during the intra-prediction process.

Utilizing ‘slices’ is a popular method to achieve parallelism for H.264 or HEVC bitstreams, as this approach eliminates the dependency on neighbors at the slice boundary. However, dividing a picture into multiple slices reduces the compression efficiency due to the requirement of slice headers, and this might not suited for low bitrate applications. Also, the ramp-up period to reach peak block processing rate is still not eliminated even with 8 or 17 slices per picture for 1080p video content. Aspects of the present disclosure include two proposed methods of encoding bitstreams that allow efficient processing of Intra-prediction on the GPU during the decoding process, even for single-slice bitstreams. These methods achieve peak block processing rate on the GPU throughout the execution and reduces the number of synchronization points needed, even for worst-case dependency.

Another reason for dividing a video picture into multiple slices is for better error resilience. When single-slice bitstreams are used, any error in the residual coefficient data of I-picture results in incorrect decoded data starting from the block containing the error to the last block in the picture. With multiple-slices, the incorrect decoded data due to the error can be limited to the end of the slice. The proposed methods could be used for creating many sections within a picture that could be decoded independently of each other, while requiring fewer slices. For an I-picture configuration that requires a specific number of independent sections or a specific number of blocks per independent section, current methods use a different slice for each independent section. This has the drawback of requiring a significant number of extra bits due to inefficient prediction and slice header overhead. A method of achieving similar number of independent sections with better prediction and fewer bits has been proposed.

For H.264 the common approach to improve parallelism is to utilize multiple slices, where each slice can be encoded or decoded virtually independent of other slices. But each slice reduces compression efficiency due to the overhead of slice header and the increase in bits needed for macroblocks at the slice boundary. In this document, a block is used to define a basic coding unit, such as a macroblock for H.264 or a Coding Tree Block (CTB) for HEVC. The prediction unit size could be smaller than the block size.

The proposed methods could be used for creating sections of a picture for which decoding processes could be performed independently of each other, without requiring multiple-slices. This can be used to speed up the encoding and decoding processes that have dependencies on neighboring blocks, and also for improving error resilience for I-pictures. When multiple slices are used, the proposed methods can achieve much higher levels of parallelism with fewer slices compared to normal encoding and decoding.

Modern GPUs are capable of performing general purpose computations in addition to graphics tasks. Such general purpose GPU (GPGPU) code can be programmed using compute shaders, which determines the number of threads used in a thread group. A 'dispatch' function is used to execute a compute shader on the GPU, and determines the number of thread groups that are launched. Ideally, only a single dispatch is required for a compute shader without any dependencies. However, tasks such as intra-prediction need synchronization once a list of blocks have been processed, so that the data needed for processing the next batch of blocks are ready. GPUs need a large number of threads to be run

in parallel in order to perform efficiently. For intra-prediction, a large number of blocks should be processed per dispatch for efficient utilization of the GPU.

FIG. 3 shows the number of blocks that can be processed during each run of the compute shader for a 1080p picture with worst-case dependency, where the basic coding unit is assumed to be 16x16. To process the entire picture, 254 dispatches are needed which equates to a similar number of synchronization steps. Also, there are periods near the beginning and end of the decoding where only a small number of blocks are actively processed on the GPU.

The particular example shown in FIG. 3 is for decoding a single-slice frame encoded using H.264 on a GPU. The height of each spike shows number of macroblocks (MB) processed per GPU dispatch for Intra-prediction. It is noted that H.264 uses 16X16 MB size, but the Intra-prediction process could alternatively use 16X16, 8X8 or 4X4 sized partitions as the basic unit of prediction for each MB. The HEVC standard by contrast uses sections called coding tree units (CTU), which can be from 16x16 pixels up to 64X64 pixels, with smaller partition sizes for prediction units. The worst case generally involves encoding the smallest prediction unit size possible and there is data dependency on the left neighbor and upper right neighbor.

A common method to break the dependencies on neighboring blocks is to employ slices or tiles while encoding the bitstream. H.264 and HEVC standards define slices and tiles, which are sections of the picture that contain integer number of coding blocks, and can be encoded or decoded nearly independently of other slices or tiles. However, using a large number of slices/tiles reduces the compression efficiency due to the overhead of header bits and because a large number of slices creates a large number of blocks at the boundary for which the prediction will be sub-optimal. A block with sub-optimal prediction can be defined as one that will not have the option to choose from all the prediction modes available for a similar single-slice stream. Such sub-optimal blocks will have a greater number of residual coefficients, which need more bits for encoding.

While multiple slices improves parallelism, the ramp-up period to reach peak block processing rate on the GPU is still not eliminated even with 8 or 17 slices per picture for 1080p video content. According to aspects of the present disclosure, the benefits of multiple slices such as improved parallelism could be obtained without needing multiple slices, which

leads to lower slice header bits and fewer sub-optimal prediction blocks. Aspects of the present disclosure can achieve peak block processing rate on a GPU throughout the execution and reduces the number of synchronization points needed for worst-case dependency. For applications that require multiple slices per picture, these methods may help reduce the number of slices needed.

EMBODIMENTS

Aspects of the present disclosure overcome problems with image and video processing that have dependencies on neighboring blocks that arise when attempting to implement such algorithms efficiently on multi-processors. Aspects of the present disclosure may be implemented with slightly modified encoders and existing optimized decoders. Examples of existing coding standards that may be used include current versions of HEVC and H.264 and also JPEG or any other block based coder that has the same issue of dependency on neighbors and can choose which neighbors it uses to code a particular block. The pattern of dependencies typically depends on the frame resolution, which includes the aspect ratio. The number of sections that can be decoded per dispatch in parallel may be maximized by breaking up the dependencies. Encoding selected sections in an independent manner and limiting the encoding options for certain sections neighboring the selected blocks as described herein can reduce or eliminate dependencies and thereby allow for efficient encoding and decoding using parallel processing. A key feature of aspects of the present disclosure is the selection of the sections that are encoded independently.

According to aspects of the present disclosure, the selected sections that are encoded independently belong to the same list number N. To choose N, consider the example shown in FIG. 3. In this case the number of sections (e.g., blocks) processed per dispatch ramps up to a maximum and then declines. List number N may be chosen as the dispatch number where the number of sections processed per dispatch during decoding decreases after the maximum. In the example shown in FIG. 3, this is dispatch number 136. In FIG. 4A, sections in list N are encoded in an independent manner such that they can be decoded independent of any neighboring sections. These sections can be decoded in parallel without having to wait until any of their neighbor blocks are decoded. Sections in list N+1, which are to the right of the blocks in list N are encoded using modes that allow the blocks to be decoded using their left neighbors only. Examples of such modes in H.264 for 4X4 and 8X8 partition sizes include modes known as “horizontal” and “horizontal-up”. For 16x16

prediction unit size, such modes include the “horizontal” mode. Sections in list N+2, which are those below the sections in list N and to the right of those in list N+1, can be encoded with modes that do not use the above-left blocks for prediction. For 4X4 and 8X8 partition sizes, sections in list N+2 may use any mode other than “diagonal-down-right”, “vertical-right”, or “horizontal-down” in the H.264 coding standard. The modes that may be used are “vertical”, “horizontal”, “DC”, “diagonal-down-left”, “vertical-left” or “horizontal-up”. For 16x16 prediction unit size, any mode other than “plane” may be used for sections in list N+2. The modes that may be used for this case are “vertical”, “horizontal” and “DC”.

By way of example, and not by way of limitation, dependencies on neighboring blocks for H.264 Luma Intra-Prediction of 4x4 and 8x8 section sizes for different intra-prediction modes are shown in Table I below.

TABLE I

Name of Prediction Mode	Above	Above Right	Left	Above Left
Vertical	Yes			
Horizontal			Yes	
DC	If Available		If Available	
Diagonal_Down_Left	Yes	Yes		
Diagonal_Down_Right	Yes		Yes	Yes
Vertical_Right	Yes		Yes	Yes
Horizontal_Down	Yes		Yes	Yes
Vertical_Left	Yes	Yes		
Horizontal_Up			Yes	

By way of further example, and not by way of limitation, dependencies on neighboring blocks for H.264 Luma Intra-Prediction of 16x16 section sizes for different intra-prediction modes are shown in Table II below.

TABLE II

Name of Prediction Mode	Above	Left	Above Left
Vertical	Yes		
Horizontal		Yes	

DC	If Available	If Available	
Plane	Yes	Yes	Yes

Encoding digital pictures in the strategy illustrated in FIG. 4A can create bitstreams that allow parallel intra-prediction. In this method, all sections in List N are encoded in such a way that it does not predict from any neighboring sections. The sections in List N+1 only use the luma or chroma samples from the neighbor section to its left for prediction. The sections in List N+2 do not predict from the neighbor section to the above-left, but can predict from the above, left, and above-right neighbor sections. This strategy allows all sections to the right and below List N to be decoded independent of all blocks to the left and above List N. The prediction modes can be chosen according to the strategy described above by disabling certain prediction modes based on the block size, or by modifying the parameters that determine if the neighboring blocks are available. For example, in the case of H.264 4x4 intra-prediction, for the blocks in List N+1, the encoder may disable all modes except Intra_4x4_Horizontal and the Intra_4x4_Horizontal_Up prediction modes.

Two implementations are proposed that allow encoding of blocks in List N without predicting from any neighboring blocks. According to a first implementation, all blocks in List N are encoded using pulse code modulation (PCM) by directly representing the sample values without prediction or application of a transform. Both H.264 and HEVC have the concept of PCM in which no prediction is done and the encoding is lossless. This mode can be invoked to break up the data dependencies. The PCM mode is a lossless mode and blocks encoded using this mode use a large number of bits compared to other modes. This is not a problem for high bitrate applications, e.g., Blu-Ray players. This technique of achieving parallel intra-prediction using PCM, is referred to as PIPCM hereafter. PIPCM allows any optimized GPU-based decoder or multi-threaded decoder to efficiently and quickly decode bitstreams created using this technique. An additional advantage to PIPCM is that no modification is needed to the decoder in order to decode pictures encoded in this fashion.

A second implementation uses an encoding mode commonly known as the “DC” mode in intra-prediction to break the dependency on neighboring blocks. In DC the encoder calculates an average of the boundary luma or chroma samples to left of the section that is to be encoded and the boundary luma or chroma samples above the section that is to be encoded and uses the calculated average for intra-prediction. If only one of the section to the left and the section above is available, the DC mode uses the available section for encoding. If

neither the section to the left nor the section above is available the DC mode uses a fixed value (e.g., 128 for 8 bit pixels) to encode the section. Implementations that use the DC mode to independently encode selected sections is referred to as Parallel intra-prediction using DC or PIDC hereafter.

- 5 As with PIPCM, PIDC involves modification of the encoder to properly encode the pictures so that they can be efficiently decoded in parallel. Unlike PIPCM, however, PIDC also requires modification of the decoder to correctly reconstruct the decoded picture. Therefore, this method is most suitable for applications where both the encoder and decoder can be controlled and modified. Examples of such a system could be an application for which the
10 encoding and decoding are both done using the same device or related devices for which both the encoding and the decoding processes can be controlled by a common entity.

To encode pictures using PIDC, the encoder may be modified to ignore the fact that the left and above neighbors are available for sections in list N so that the sections in list N are always as encoded with a fixed value using DC mode. In order to properly decode pictures
15 encoded in this way, the decoder is also modified to ignore the fact that the above and left neighbors of the list N sections are available.

The determination of list number N may depend on the height of the picture in terms of the relevant section size and how the picture is encoded. By way of example and not by way of limitation, if the picture height in units of blocks is represented as MB_h, the value of N for
20 PIDC encoded pictures is: $N = (2 * MB_h)$, when Mb_h is even and $N = (2 * MB_h - 2)$, when Mb_h is odd. For PIPCM encoded pictures, the value of N is: $N = (2 * MB_h - 1)$, when Mb_h is even and $N = (2 * MB_h - 3)$, when Mb_h is odd.

Depending on the encoding mode (PIDC or PIPCM) mode restrictions need not necessarily be enforced for all subsections of the sections in lists N, N+1, and N+2. For PIPCM mode,
25 the H.264 standard does not allow different modes for different sub-units of a section, and so the mode is for the entire section, e.g., an entire 16x16 prediction unit. However, for in PIDC mode, while all sections in list N may be set to be encoded as DC, it may be sufficient to only enforce the DC mode restriction for the subsections of those sections (e.g., the sub-blocks or partitions in a block in list N) that are adjacent to the left or top neighbor. An example of this
30 is shown in FIG. 4B. The remaining subsections represented in white could be encoded using any available mode. For a block in list N+1, it is sufficient to enforce the mode restrictions

for the subsections shown in gray (those that are adjacent to the left or top neighbor). Within the sections containing those subsections, the subsections represented in white, could be encoded using any mode. Similarly, for a block in list N+2, it is sufficient to enforce the mode restrictions for the subsections shown with the diagonal pattern (those that are adjacent to the left or top neighbor). The subsections represented in white, could be encoded using any mode.

The encoder may decide how to partition each block in order to minimize the prediction error, which in turn minimizes the number of bits needed to encode the residual data. The encoder may use a metric such as Sum of Absolute Differences (SAD) to determine which partition size would minimize the number of bits needed to encode the block. By way of example, and not by way of limitation, if a 16x16 block has a flat area with no fine detail, encoding the block entirely as 16x16 may result in the least bits used. Alternatively, if there is a lot of fine detail in a 16x16 block, then splitting it into smaller sized sub-units (e.g., 8x8 sub-blocks or 4x4 partitions) and using different modes for two or more of the smaller sized sub-unit may reduce or minimize the number of bits needed to encode the block.

In example illustrated in FIG. 4B, it is assumed that each block is 16x16, although aspects of the present disclosure are not limited to such implementations. If the entire block is shaded, the block has been encoded using a 16x16 Intra-prediction mode. If three of the four sub-blocks are shaded, it represents 8x8 Intra-prediction. If seven of the sixteen 4x4 partitions are shaded, it represents 4x4 Intra-prediction.

It is noted that even if a section (e.g., a 16x16 block) in list N is split into smaller sized sub-units (e.g., 8x8 sub-blocks or 4x4 partitions), the section as a whole can be encoded independently of any of its neighboring sections (e.g., neighboring 16x16 blocks). Similarly, if a section (e.g., a 16x16 block) in list N+1 is split into smaller sized sub-units, the section as a whole can be encoded using at most the section to its immediate left. Likewise a section (e.g., a 16x16 block) in list N+2 is split into smaller sized sub-units (e.g., 8x8 sub-blocks or 4x4 partitions), the section as a whole can be encoded using at most the sections to its immediate left, immediately above it and immediately to its above-right.

FIG. 5 provides an illustrative example of parallel intra-prediction encoding according to an aspect of the present disclosure. In particular, FIG. 5 depicts an example process flow 500 for encoding a digital picture using parallel intra-prediction encoding in accordance with certain

implementations of the present disclosure. It is important to note that the example of FIG. 5 is only a simplified example for purposes of illustrating only certain aspects of how a digital picture may be encoded in accordance with the present disclosure.

At the outset, it is important to note that in this example, a digital picture is undergoing the encoding process via the activity of a system utilizing PIPCM and/or PIDC. In this example, as shown in FIG. 5, intra-prediction is started **501** on the digital picture. In certain implementations, the system may utilize PIPCM to encode the sections in list number (N). In alternative implementations, the system may utilize PIDC to encode the sections in list number (N). Depending on a variety of factors described above, an appropriate encoding method may be chosen **510**. By way of example, and not by way of limitation, one such factor is whether or not it is known that the picture being encoded will be decoded by a decoder that is compatible with PIDC. In some cases this information may be known beforehand. In other cases it may be necessary for the system performing the encoding to query the system that performs the decoding in order to determine whether the decoder is compatible with PIDC. In some situations, the query and its response may be transmitted over a network, which may be a personal area network, local area network, wide area network, wireless network, fiber optic network, satellite transmission network or other network between two remotely located systems.

In certain embodiments, PIDC may be utilized, and the system may identify the list number (N) accordingly **512**. The identification of list number (N) is done for maximum parallelism, and as an example, can be based on video resolution, or in alternative embodiments, (N) may be defined based on a predetermined rule set known to both the encoder and decoder utilized. For all blocks in list N, neighboring blocks are marked unavailable, and the blocks are encoded **516** using a valid mode (e.g., DC) based on modified neighbor information. Subsequently, the blocks in list N+1 are encoded **520** using only prediction modes that do not predict from the Above-Left, Above, and Above-Right blocks. Blocks in list N+2 are encoded **522** using only prediction modes that do not predict from the Above-Left blocks. Finally, all other blocks in the picture are encoded **524** using the best intra-prediction mode.

In alternative embodiments, wherein PIDC is not utilized, the system may identify list number (N) accordingly **514**. The identification of list number (N) is done for maximum parallelism, and as an example, can be based on video resolution, or in alternative embodiments, (N) may be defined based on a predetermined rule set. All blocks in list N have

at least some subsections thereof encoded **520** using a mode that does not predict from neighbors, e.g. PCM. Subsequently, the blocks in list N+1 have at least some subsections thereof encoded **520** using only prediction modes that do not predict from the Above-Left, Above, and Above-Right blocks. Blocks in list N+2 have at least some sub-sections encoded
5 **522** using only prediction modes that do not predict from the Above-Left blocks. Finally, all other blocks in the picture are encoded **524** using the best intra-prediction mode.

Turning now to FIG. 6, an illustrative example of parallel intra-prediction decoding is provided. In particular, FIG. 6 depicts an example process flow **600** for decoding a digital picture using parallel intra-prediction decoding in accordance with certain implementations
10 of the present disclosure. It is important to note that the example of FIG. 6 is only a simplified example for purposes of illustrating only certain aspects of how a digital picture may be decoded in accordance with the present disclosure.

At the outset, it is important to note that in this example, a digital picture is undergoing the decoding process via the activity of a system utilizing PIPCM and/or PIDC. In this example,
15 as shown in FIG. 6, intra-prediction is started **601** on the digital picture. In alternative implementations, the picture may have been encoded using PIDC, in which case the system may utilize PIDC to decode the picture. Depending on a variety of factors described above, an appropriate decoding method is chosen **610**.

In some cases the type of encoding for a picture (e.g., PIPCM or PIDC) may be known to the decoding system beforehand. In other cases it may be useful for the system performing the
20 encoding to notify the system that performs the decoding of the type of encoding used for the picture in order for the decoder to select an appropriate decoding method. In some situations, the notification may be transmitted over a network, which may be a personal area network, local area network, wide area network, wireless network, fiber optic network, satellite
25 transmission network or other network between two remotely located systems. In some implementations, the notification may be included in a data packet containing part of the bitstream for the picture or in a header for such a data packet.

In certain embodiments, PIDC may be used, and the system may define list number (N) accordingly **612**. The definition of list number (N) can be based, for example, on a
30 predetermined rule set known to both the encoder and decoder utilized. Alternatively, list number (N) may be defined from information in the encoded bitstream. For all blocks in list

N, neighboring blocks are marked unavailable **614**. Subsequently, the blocks are grouped **616** into lists based on the modified dependency of the blocks on their neighboring blocks. Each list is then dispatched **618** to the GPU and synchronization is performed after each list is complete.

- 5 In alternative embodiments, wherein PIDC is not utilized, the system groups the blocks **616** into lists based on the dependency of the blocks on their neighboring blocks. Each list is then dispatched **618** to the GPU and synchronization is performed after each list is complete.

As shown in the example above, in PIDC, the blocks in List N are encoded using the DC mode, and the method to determine the availability of neighboring blocks is modified.

- 10 According to the H.264 and HEVC standards, the DC mode utilizes neighboring luma or chroma samples for prediction unless the neighboring blocks are unavailable. But neighboring blocks are unavailable for prediction only for the first block of a slice, and are available for other blocks. By utilizing the DC mode for prediction and setting the neighboring blocks as unavailable, even if they are available in reality, the encoder forces a
15 break in the dependency of the sections in list N on neighboring sections. Without modifying the decoder, the reconstructed data will not be accurate, as the decoder assumes that the neighbors are available for the blocks in List N and uses the neighbors for prediction. Therefore, in order to ensure accuracy of the decoded picture, both the encoder and the decoder should follow the same rules to decide the value of N, and also follow the same
20 strategy for determining the availability of neighbors for all blocks in List N.

- The value of N is selected based on the requirement of the application. For fast decoding of 1080p streams, a value of 135 could be chosen for N when the PIPCM method is used, and a value of 136 could be chosen for N for the PIDC method. With these values, the number of blocks that can be processed during each run of a compute shader is shown in FIG. 7, for
25 worst-case dependency. The number of dispatches required is lower compared to the results shown in FIG. 3 for a conventional encoding algorithm, and the peak block processing rate is achieved for every dispatch without any ramp up period, which leads to good load balancing.

- For ideal load balancing, the value of N depends on the picture height. For even faster decoding and error resilience, there could be multiple values of N according to the speedup
30 required or number of independent sections needed. By way of example and not by way of

limitation, a faster variation of PIDC for 1080p may encode lists N, $2*N$, and $3*N$ as DC, where $N = 68$.

FIG. 8 shows the mode selection process when PIPCM mode is used to encode a 1080p picture, and the value of N chosen is 135. The blocks coded as PCM are shaded as black.

5 Similarly, FIG. 9 shows the mode selection process when PIDC is used for encoding and the value of N is selected as 136. The blocks that do not have any dependency on neighbors due to the modified DC mode are indicated in black. For both FIG. 8 and FIG. 9, the blocks in light grey only use luma or chroma samples from the block to its left for prediction, and the blocks in medium gray do not use luma or chroma samples from the block to its above-left
10 for prediction.

Besides applications in speeding up the encoding and decoding processes, parallel intra-prediction could also be used for improving error resilience for I-pictures. When single-slice bitstreams are used, any error in the I-picture results in incorrect decoded data starting from the block containing the error to the last block in the picture. With multiple-slices, the
15 incorrect decoded data due to the error can be limited to the end of the slice. Typical multi-slice configurations for error resilience utilize either a fixed number of blocks per slice, or a fixed number of bits per slice. An example of a 36-slice 720p picture **700** with 100 blocks in each slice **710** is shown in FIG. 10. This arrangement ensures that there are groups of 100 blocks that can be encoded and decoded independent of each other, which constrains any
20 error in a section from crossing the boundary of the section. The disadvantages of using such a large number of slices are the overhead of slice headers and the large number of blocks with sub-optimal prediction.

Slices and tiles are used by H.264 and HEVC codecs to parallelize encoding and decoding processes. For faster encoding and decoding, a large number of slices could be used based on
25 the video resolution (8-20 slices for 1080p content that is not streamed over the network). But increasing the number of slices decreases the compression efficiency, so it is a trade-off. Slices are also used for error concealment to limit the number of sections that need to be concealed in case of error.

FIG. 11 shows an example of how parallel intra-prediction could be used to create 36
30 independently decodable segments **760** of 100 blocks each using only 9 slices **770** of a picture **750**. This could be used for better error concealment for applications such as a Blu-

Ray disc player. For cases where blocks from different lists are encoded as PCM/DC, we may use the block number in raster scan order or the position of the block in terms of row and column index to determine the location of the PCM/DC blocks. If the information about the blocks that are coded as PCM/DC is not included in the bitstream, the rules to determine the position of these blocks should be predetermined and known to the decoder.

Slices and tiles are defined in H.264 and HEVC standards and are used to parallelize encoding and decoding processes. Slices and tiles are groups of macroblocks/CTBs for which a majority of encoding and decoding tasks could be performed without any dependency on macroblocks/CTBs that do not belong to the slice or tile. The term 'segment' is used to refer to the groupings of blocks or other sections shown in FIG. 11 and FIG. 12 because these groups of blocks share this property of slices and tiles, but cannot be classified as slices or tiles according to the definition defined in the standard.

For dynamic determination of the position of the PCM/DC blocks, the required information for correct decoding could be stored in the bitstream as supplemental enhancement information (SEI). By using the parallel intra-prediction method, the decoder will be able to create a map of independently decodable segments using the position information of the PCM/DC blocks, and perform error concealment only for the segment containing an error instead of the entire slice, when the error is encountered during decoding of residual or predicted luma or chroma samples.

By defining a flag 'horizontal_run_mode' that when enabled, controls the prediction mode for all blocks between two successive PCM/DC modes, we could further reduce the number of slices needed for creating sections that are independent of each other. FIG. 12 shows how parallel intra-prediction and horizontal_run_mode could be used to create 36 independently decodable segments **810** of 100 blocks each using only 1 slice **820** of a picture **800**.

Turning now to FIG. 13, an illustrative example of a system **900** configured to operate in accordance with aspects of the present disclosure is depicted. According to aspects of the present disclosure, the system **900** may be an embedded system, mobile phone, personal computer, tablet computer, portable game device, workstation, game console, and the like.

The system **900** generally includes a central processor unit (CPU) **920** which may include a CPU core and other features of the type discussed above. By way of example and not by way of limitation, the CPU **920** may be part of an accelerated processing unit (APU) **910** that

includes the CPU **920**, and a graphics processing unit (GPU) **930** on a single chip. In alternative implementations, the CPU **920** and GPU **930** may be implemented as separate hardware components on separate chips.

The system **900** may also include memory **940**. The memory **940** may optionally include a
5 main memory unit that is accessible to the CPU **920** and GPU **930**, and portions of the main memory may optionally include portions of the graphics memory **942**. The CPU **920** and GPU **930** may each include one or more processor cores, e.g., a single core, two cores, four cores, eight cores, or more. The CPU **920** and GPU **930** may be configured to access one or more memory units using a data bus **990**, and, in some implementations, it may be useful for
10 the system **900** to include two or more different buses.

The memory **940** may include one or more memory units in the form of integrated circuits that provides addressable memory, e.g., RAM, DRAM, and the like. The memory may contain executable instructions configured to implement a method for encoding and/or decoding a picture in accordance with the embodiments described above. The graphics
15 memory **942** may temporarily store graphics resources, graphics buffers, and other graphics data for a graphics rendering pipeline. The graphics buffers may include, e.g., one or more vertex buffers for storing vertex parameter values and one or more index buffers for storing vertex indices. The graphics buffers may also include a one or more render targets **944**, which may include both color buffers **945** and depth buffers **946** holding pixel/sample values
20 computed as a result of execution of instructions by the CPU **920** and GPU **930**. In certain implementations, the color buffers **945** and/or depth buffers **946** may be used to determine a final array of display pixel color values to be stored in a display buffer **947**, which may make up a final rendered image intended for presentation on a display. In certain implementations, the display buffer may include a front buffer and one or more back buffers, and the GPU **930**
25 may be configured to scanout graphics frames from the front buffer of the display buffer **947** for presentation on a display **980**.

The CPU **920** may be configured to execute CPU code, which may include operating system (OS) **921** or an application **922** (e.g., a video game). The OS **921** may be configured to implement certain features of operating the CPU **920** as discussed above. The CPU code may
30 include a graphics application programming interface (API) **924** for issuing draw commands or draw calls to programs implemented by the GPU **930** based on a state of the application **922**. The CPU code may also implement physics simulations and other functions. In some

implementations, the CPU **920** may include an encoder **927** or decoder **929** configured to implement video encoding and decoding tasks including, but not limited to, encoding and/or decoding a picture in accordance with the embodiments described above. In an alternative implementation, the encoder **927** and decoder **929** may be part of a video coding engine that includes both the encoder and decoder, wherein the video coding engine may be implemented in hardware, in software, or in some combination of hardware and software. The encoder **927** and decoder **929** may be respectively configured to implement video encoding and decoding tasks including, but not limited to, encoding and/or decoding a picture in accordance with the embodiments described above. In an alternative implementation, the encoder and decoder may be part of a video coding engine that includes both the encoder and decoder, wherein the video coding engine may be implemented in hardware, in software, or in some combination of hardware and software. In hardware implementations the video coding engine hardware may be part of the CPU **920**, part of the GPU **930**, a separate hardware component of the APU **910** or a separate hardware component from the APU, CPU, and GPU. In some software implementations the encoder **927** and decoder **929** may be partly or wholly implemented by execution of coded instructions with the GPU **930**.

To support the rendering of graphics, the GPU **930** may execute shaders **933**, which may include vertex shaders and pixel shaders. The GPU **930** may also execute other shader programs, such as, e.g., geometry shaders, tessellation shaders, compute shaders, and the like.

The GPU may also include specialized hardware modules **931**, which may include one or more texture mapping units and/or other hardware modules configured to implement operations at one or more stages of a graphics pipeline, which may be fixed function operations. The shaders **933** and hardware modules **931** may interface with data in the memory **940** and the buffers **944** at various stages in the pipeline before the final pixel values are output to a display **980**. The GPU **930** may include a rasterizer module **935**, which may be optionally embodied in a hardware module **931** of the GPU, a shader **933**, or a combination thereof. The rasterization module **935** may be configured take multiple samples of primitives for screen space pixels and invoke one or more pixel shaders according to the nature of the samples.

The system **900** may also include well-known support functions **950**, which may communicate with other components of the system, e.g., via the bus **990**. Such support functions may include, but are not limited to, input/output (I/O) elements **952**, one or more

clocks **956**, which may include separate clocks for the CPU and GPU, respectively, and one or more levels of cache **958**, which may be external to the CPU **920**. The system **900** may optionally include a mass storage device **960** such as a disk drive, CD-ROM drive, flash memory, tape drive, Blu-ray drive, or the like to store programs and/or data. In one example, the mass storage device **960** may receive a computer readable medium **962** containing video data to be encoded and/or decoded. Alternatively, the application **962** (or portions thereof) may be stored in memory **940** or partly in the cache **958**.

The device **900** may also include a display unit **980** to present rendered graphics **982** prepared by the GPU **930** to a user. The device **900** may also include a user interface unit **970** to facilitate interaction between the system **900** and a user. The display unit **980** may be in the form of a flat panel display, cathode ray tube (CRT) screen, touch screen, head mounted display (HMD) or other device that can display text, numerals, graphical symbols, or images. The display **980** may display rendered graphics **982** processed in accordance with various techniques described herein. The user interface **970** may contain one or more peripherals, such as a keyboard, mouse, joystick, light pen, game controller, touch screen, and/or other device that may be used in conjunction with a graphical user interface (GUI). In certain implementations, the state of the application **922** and the underlying content of the graphics may be determined at least in part by user input through the user interface **970**, e.g., where the application **922** includes a video game or other graphics intensive application.

The system **900** may also include a network interface **972** to enable the device to communicate with other devices over a network. The network may be, e.g., a local area network (LAN), a wide area network such as the internet, a personal area network, such as a Bluetooth network or other type of network. Various ones of the components shown and described may be implemented in hardware, software, or firmware, or some combination of two or more of these.

Aspects of the present disclosure facilitate parallelization of encoding and decoding of digital pictures in a way that can be implemented in a fairly straightforward manner with modified versions of existing codec software or hardware. In some implementations, no modification is required on the decoder side. In others the modifications are straightforward.

While the above is a complete description of the preferred embodiment of the present invention, it is possible to use various alternatives, modifications and equivalents. Therefore, the scope of the present invention should be determined not with reference to the above

description but should, instead, be determined with reference to the appended claims, along with their full scope of equivalents. Any feature described herein, whether preferred or not, may be combined with any other feature described herein, whether preferred or not. In the claims that follow, the indefinite article “A”, or “An” refers to a quantity of one or more of

5 the item following the article, except where expressly stated otherwise. As used herein, in a listing of elements in the alternative, the word “or” is used in the logical inclusive sense, e.g., “X or Y” covers X alone, Y alone, or both X and Y together, except where expressly stated otherwise. Two or more elements listed as alternatives may be combined together. The appended claims are not to be interpreted as including means-plus-function limitations, unless

10 such a limitation is explicitly recited in a given claim using the phrase “means for.”

WHAT IS CLAIMED IS:

- 1 1. A method, comprising:
2 encoding selected sections of a digital picture in a first list, wherein each section in the
3 first list is encoded independently of encoding any of its neighbors;
4 encoding selected sections of the digital picture in a second list, wherein each section in
5 the second list is located to the right of a corresponding section in the first list and
6 wherein each section in the second list is encoded using at most the section to its
7 immediate left;
8 encoding selected sections of the digital picture in a third list, wherein each section in the
9 third list is located to the right of a corresponding section in the second list and
10 immediately below a corresponding section in the first list, wherein each section in the
11 third list is encoded using at most the sections to its immediate left, immediately above it
12 and immediately to its above-right; and
13 encoding all other sections of the digital picture.
- 1 2. The method of claim 1, wherein each section in the first list is encoded using pulse code
2 modulation.
- 1 3. The method of claim 1, wherein each section of the first list is encoded using a mode that
2 encodes a section as a fixed value when certain neighboring blocks are unavailable.
- 1 4. The method of claim 3, wherein encoding the selected sections of the digital picture in the
2 first list includes designating the certain neighboring sections of each block in the first list
3 as unavailable.
- 1 5. The method of claim 1, further comprising dividing the digital picture into two or more
2 slices.
- 1 6. The method of claim 5, wherein the sections of the first, second, and third lists are
2 selected to divide the picture into two or more groups of sections that are decodable in
3 parallel.
- 1 7. The method of claim 1, wherein the digital picture is encoded as a single slice.

- 1 8. The method of claim 7, wherein the sections of the first, second, and third lists are
2 selected to divide the picture into two or more groups of sections that are decodable in
3 parallel.
- 1 9. The method of claim 1, wherein encoding the selected sections of the digital picture in the
2 first, second, or third list includes dividing one or more selected sections in the first,
3 second, or third list into smaller sized sub-units and encoding two or more of the smaller
4 sized sub-units in different modes selected to reduce or minimize a number of bits
5 required to encode the one or more selected sections.
- 1 10. A system, comprising:
2 a processor;
3 a memory; and
4 processor executable instructions embodied in the memory, the instructions being
5 configured to implement a method upon execution by the processor, the method
6 comprising:
7 a) encoding selected sections of a digital picture in a first list, wherein each section in the
8 first list is encoded independently of encoding of any of its neighbors;
9 b) encoding selected sections of the digital picture in a second list, wherein each section
10 in the second list is located to the right of a corresponding section in the first list and
11 wherein each section in the second list is encoded using at most the section to its
12 immediate left;
13 c) encoding selected sections of the digital picture in a third list, wherein each section in
14 the third list is located to the right of a corresponding section in the second list and
15 immediately below a corresponding section in the first list, wherein each section in the
16 third list is encoded using at most the sections to its immediate left, immediately above it
17 and immediately to its above-right;
18 d) and encoding all other sections of the digital picture.
- 1 11. A non-transitory computer readable medium having computer readable instructions
2 embodied therein, the instructions being configured to implement a method upon
3 execution by a processor, the method comprising:
4 a) encoding selected sections of a digital picture in a first list, wherein each section in the
5 first list is encoded independently of encoding of any of its neighbors;

6 b) encoding selected sections of the digital picture in a second list, wherein each section
7 in the second list is located to the right of a corresponding section in the first list and
8 wherein each section in the second list is encoded using at most the section to its
9 immediate left;

10 c) encoding selected sections of the digital picture in a third list, wherein each section in
11 the third list is located to the right of a corresponding section in the second list and
12 immediately below a corresponding section in the first list, wherein each section in the
13 third list is encoded using at most the sections to its immediate left, immediately above it
14 and immediately to its above-right;

15 d) and encoding all other sections of the digital picture.

1 12. A method, comprising:

2 designating as unavailable certain neighbor sections of sections in a first list of encoded
3 sections of an encoded digital picture with a decoder even if some of the certain neighbor
4 sections are available when each section in the first list has been encoded in a mode that
5 encodes a section using a fixed value when certain neighboring sections are unavailable;
6 decoding the sections in the first list of encoded sections of the digital picture in parallel
7 with the decoder without using the neighbor sections that have been designated as
8 unavailable.

1 13. The method of claim 12, wherein selected sections of the digital picture in a second list
2 are encoded using at most the section to its immediate left, wherein each section in the
3 second list is located to the right of a corresponding section in the first list, the method
4 further comprising,

5 decoding the sections in the second list in parallel, wherein each section in the second
6 list is decoded using at most the section to its immediate left.

1 14. The method of claim 12, wherein selected sections of the digital picture in a second list
2 are encoded using at most the section to its immediate left, wherein each section in the
3 second list is located to the right of a corresponding section in the first list and wherein
4 selected sections of the digital picture in a third list are encoded using at most the sections
5 to its immediate left, immediately above it and immediately to its above-right, wherein
6 each section in the third list is located to the right of a corresponding section in the second
7 list and immediately below a corresponding section in the first list, the method further
8 comprising,

9 decoding the sections in the second list in parallel with the decoder, wherein each
10 section in the second list is decoded using at most the section to its immediate left; and
11 decoding each section in the third list in parallel with the decoder, wherein each
12 section in the third list is decoded using at most the sections to its immediate left,
13 immediately above it and immediately to its above-right.

1 15. The method of claim 12, wherein one or more selected sections in the first list are
2 divided into smaller sized sub-units, wherein two or more of the sub-units have been encoded
3 in different modes selected to reduce or minimize a number of bits required to encode the one
4 or more selected sections

1 16. A system, comprising:
2 a processor;
3 a memory; and
4 processor executable instructions embodied in the memory, the instructions being configured
5 to implement a method upon execution by the processor, the method comprising:
6 designating as unavailable certain neighbor sections of sections in a first list of encoded
7 sections of an encoded digital picture with a decoder even if some of the certain neighbor
8 sections are available when each section in the first list has been encoded in a mode that
9 encodes a section using a fixed value when certain neighboring sections are unavailable;
10 decoding the sections in the first list of encoded sections of the digital picture in parallel with
11 the decoder without using the neighbor sections that have been designated as unavailable.

1 17. The system of claim 16, wherein when selected sections of the digital picture in a second
2 list are encoded using at most the section to its immediate left and each section in the
3 second list is located to the right of a corresponding section in the first list, the method
4 further comprises,
5 decoding the sections in the second list in parallel with the decoder, wherein
6 each section in the second list is decoded using at most the section to its immediate left.

1 18. The system of claim 16, wherein when selected sections of the digital picture in a second
2 list are encoded using at most the section to its immediate left and each section in the second
3 list is located to the right of a corresponding section in the first list and when selected
4 sections of the digital picture in a third list are encoded using at most the sections to its
5 immediate left, immediately above it and immediately to its above-right and when each

6 section in the third list is located to the right of a corresponding section in the second list and
7 immediately below a corresponding section in the first list, the method further comprises,
8 decoding the sections in the second list in parallel with the decoder, wherein each section
9 in the second list is decoded using at most the section to its immediate left; and
10 decoding each section in the third list in parallel with the decoder, wherein each section in
11 the third list is decoded using at most the sections to its immediate left, immediately above it
12 and immediately to its above-right.

1 19. A non-transitory computer readable medium having computer readable instructions
2 embodied therein, the instructions being configured to implement a method upon execution
3 by a processor, the method comprising:
4 designating as unavailable certain neighbor sections of sections in a first list of encoded
5 sections of an encoded digital picture with a decoder even if some of the certain neighbor
6 sections are available when each section in the first list has been encoded in a mode that
7 encodes a section using a fixed value when certain neighboring sections are unavailable;
8 decoding the sections in the first list of encoded sections of the digital picture with the
9 decoder in parallel without using the neighbor sections that have been designated as
10 unavailable.

1 20. The non-transitory computer readable medium of claim 19, wherein when selected
2 sections of the digital picture in a second list are encoded using at most the section to its
3 immediate left and each section in the second list is located to the right of a corresponding
4 section in the first list, the method further comprises,
5 decoding the sections in the second list in parallel with the decoder, wherein
6 each section in the second list is decoded using at most the section to its immediate left.

1 21. The non-transitory computer readable medium of claim 19, wherein when selected
2 sections of the digital picture in a second list are encoded using at most the section to its
3 immediate left and each section in the second list is located to the right of a corresponding
4 section in the first list and when selected sections of the digital picture in a third list are
5 encoded using at most the sections to its immediate left, immediately above it and
6 immediately to its above-right and when each section in the third list is located to the right of
7 a corresponding section in the second list and immediately below a corresponding section in
8 the first list, the method further comprises,
9 decoding the sections in the second list in parallel with the decoder, wherein each section

10 in the second list is decoded using at most the section to its immediate left; and
11 decoding each section in the third list in parallel with the decoder, wherein each section in
12 the third list is decoded using at most the sections to its immediate left, immediately above it
13 and immediately to its above-right.

1

1

1

1/12

Above Left MB	Above MB	Above Right MB
Left MB	Curr MB	

FIG. 1A

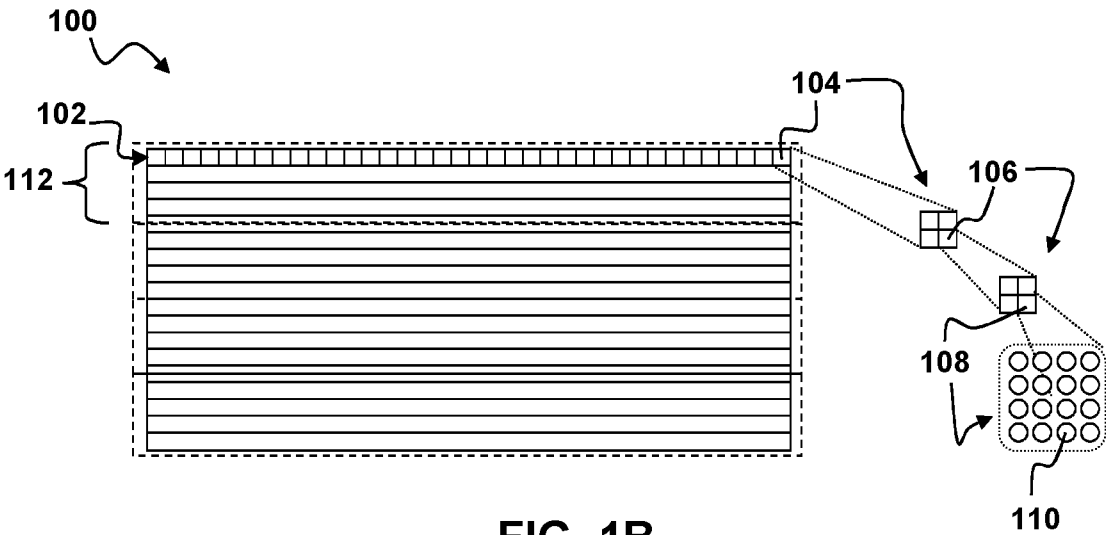


FIG. 1B

2/12

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

FIG. 2

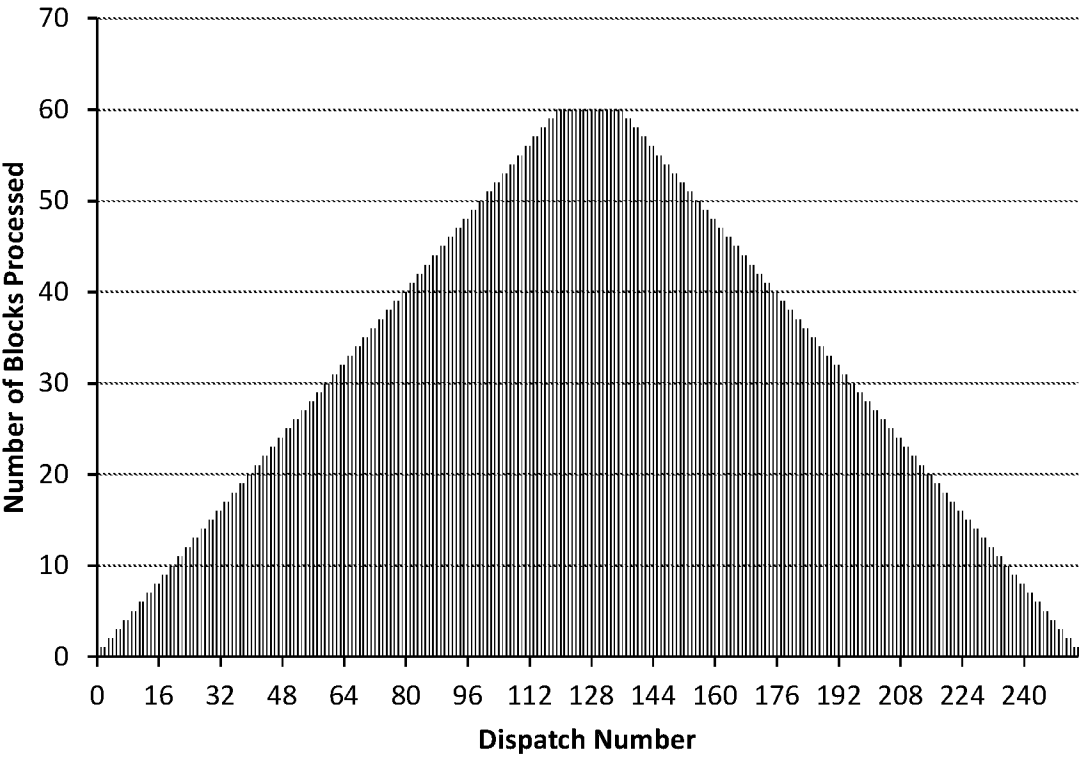


FIG. 3

4/12

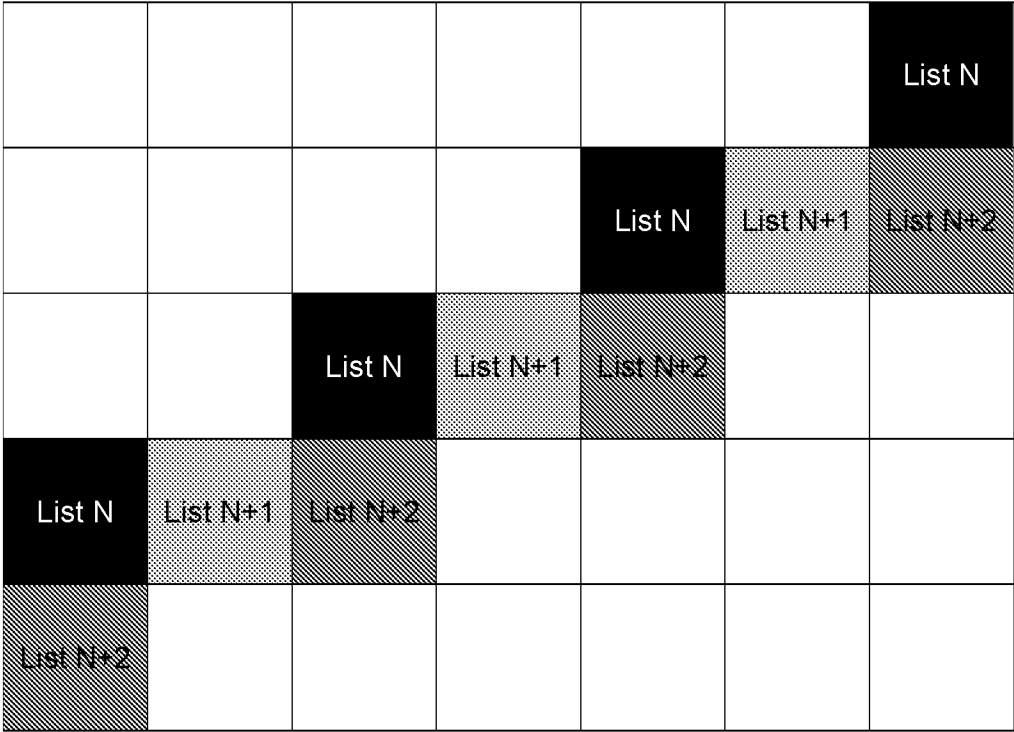


FIG. 4A

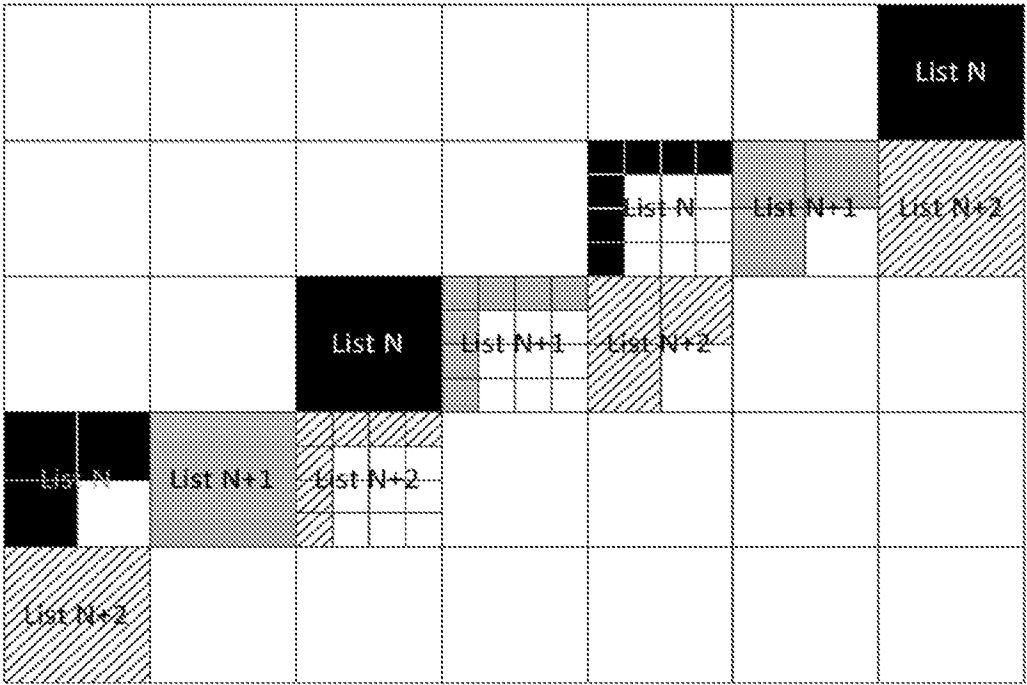


FIG. 4B

5/12

500

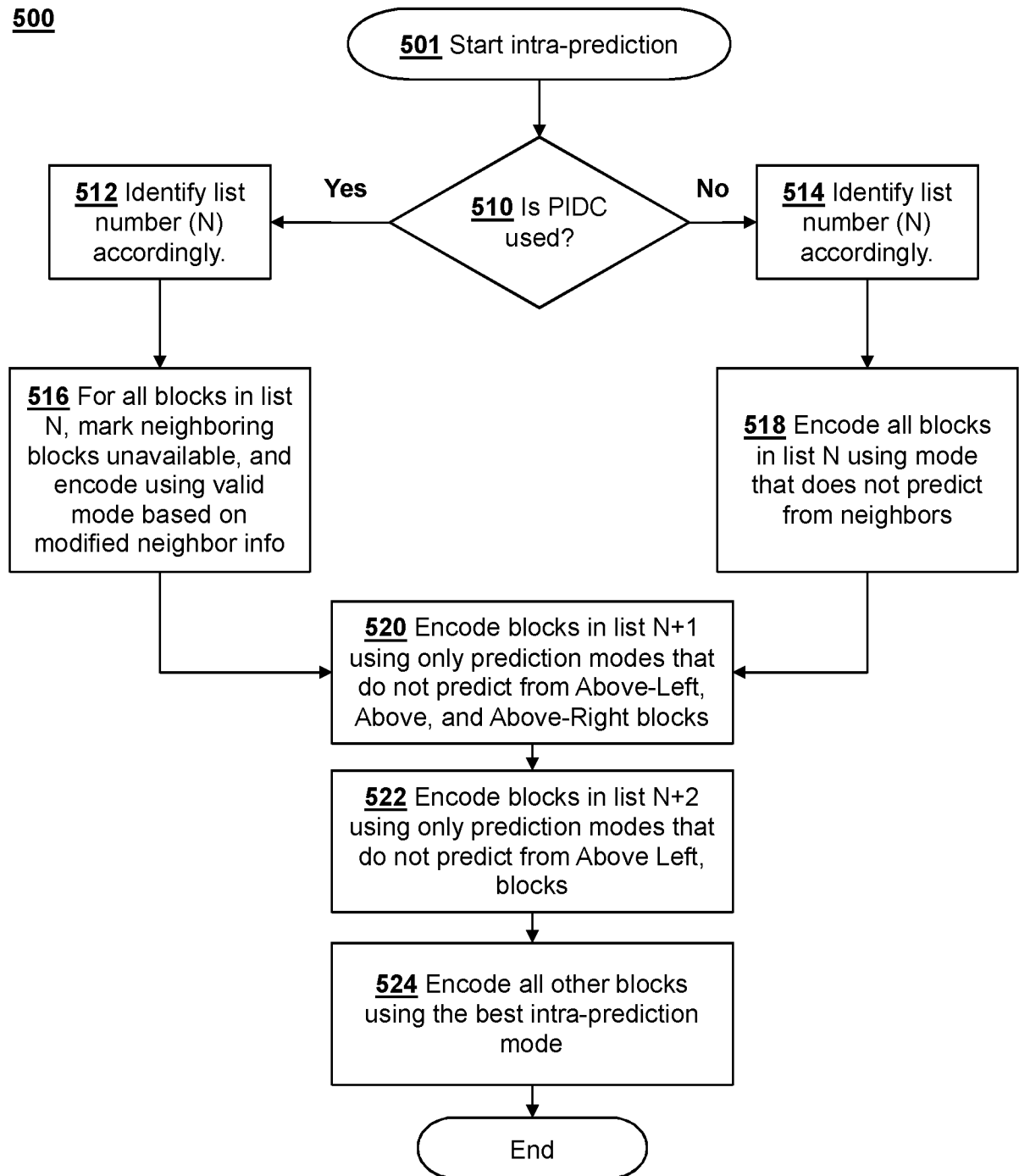


FIG. 5

6/12

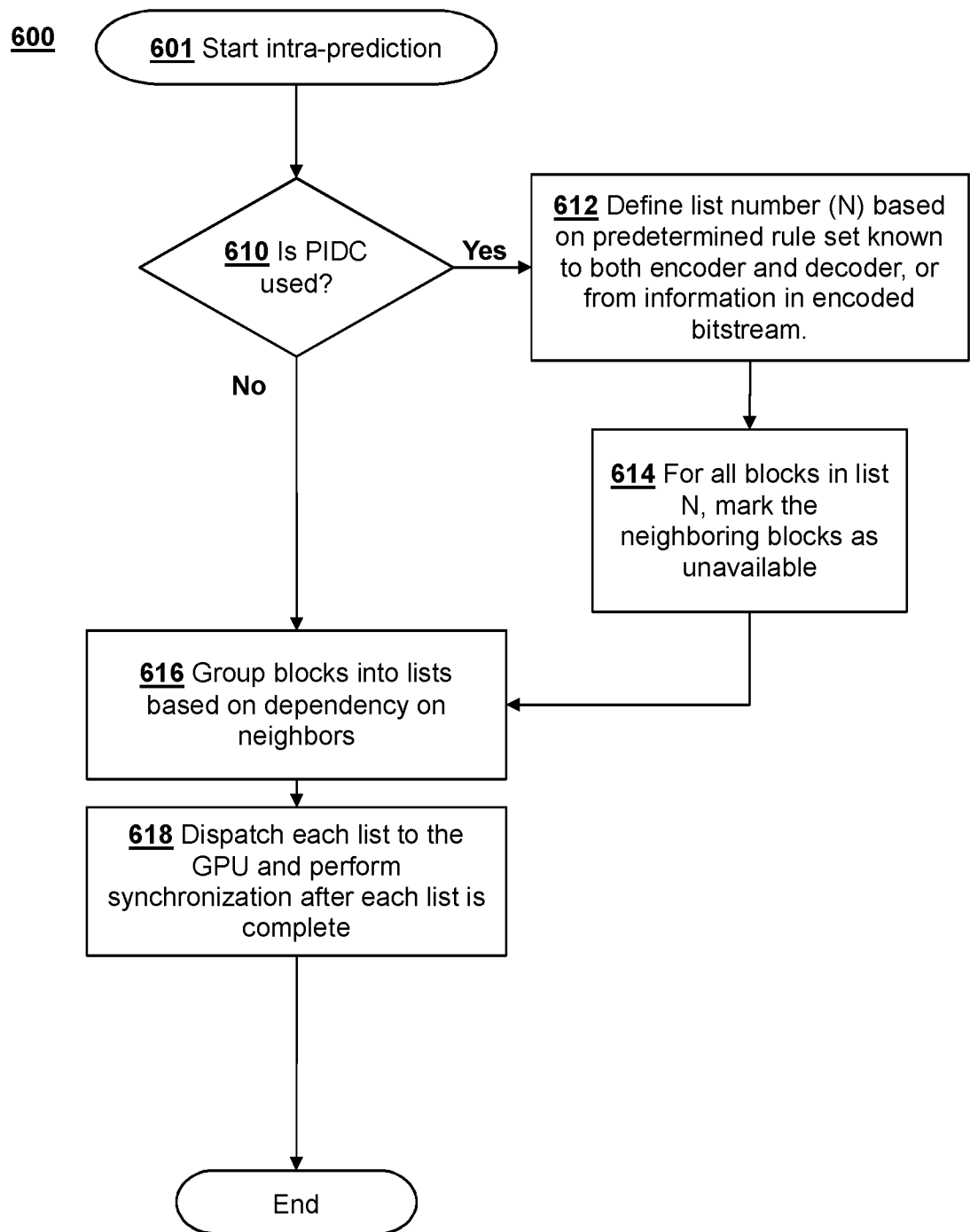


FIG. 6

7/12

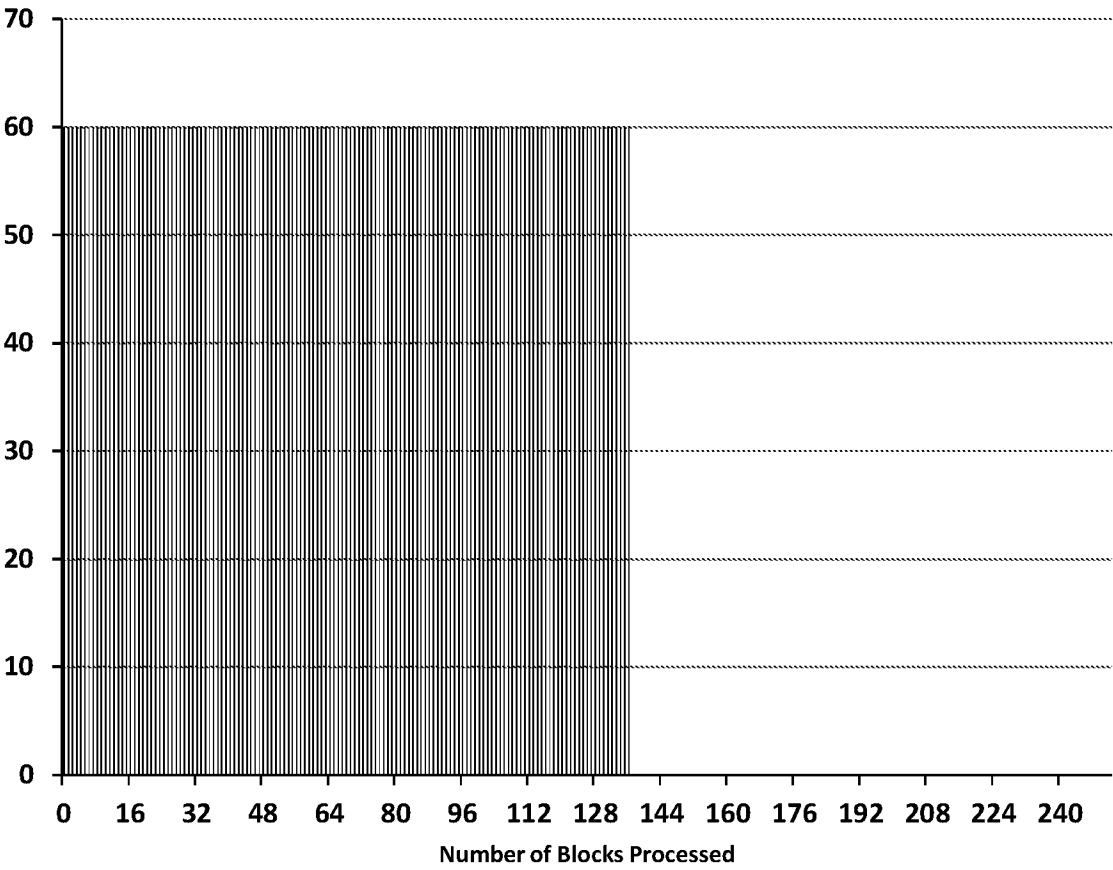


FIG. 7

8/12

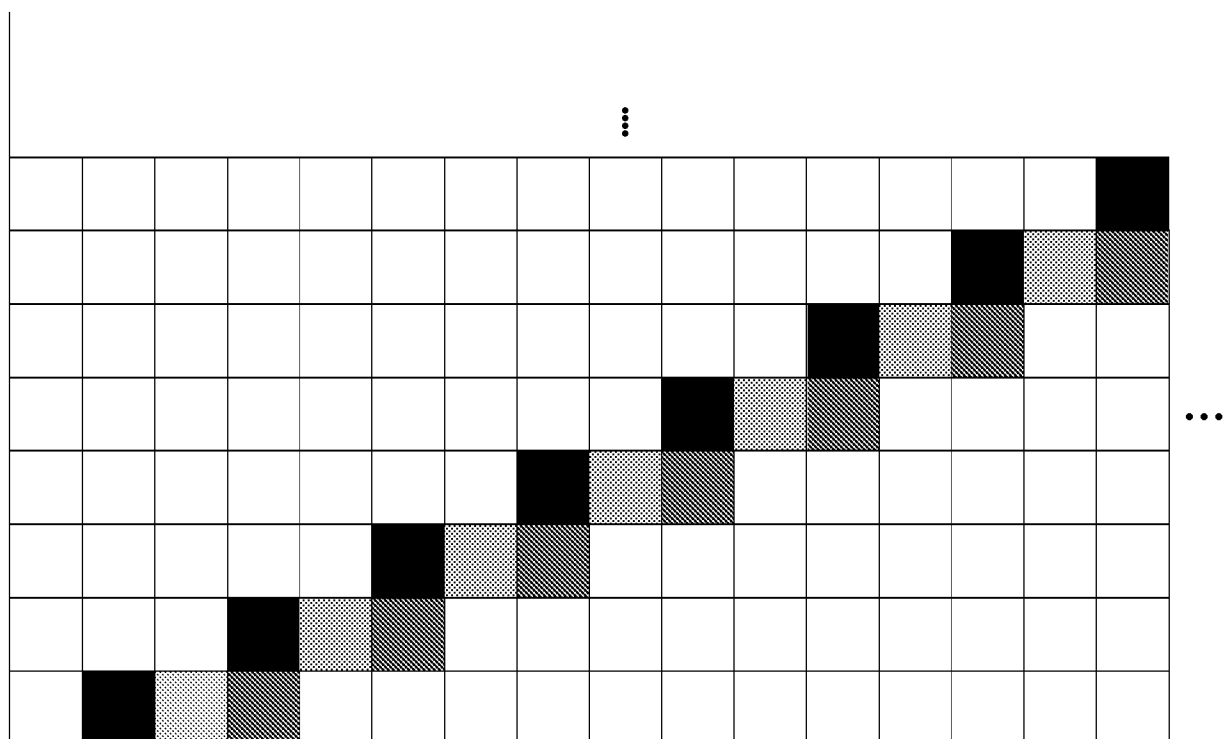


FIG. 8

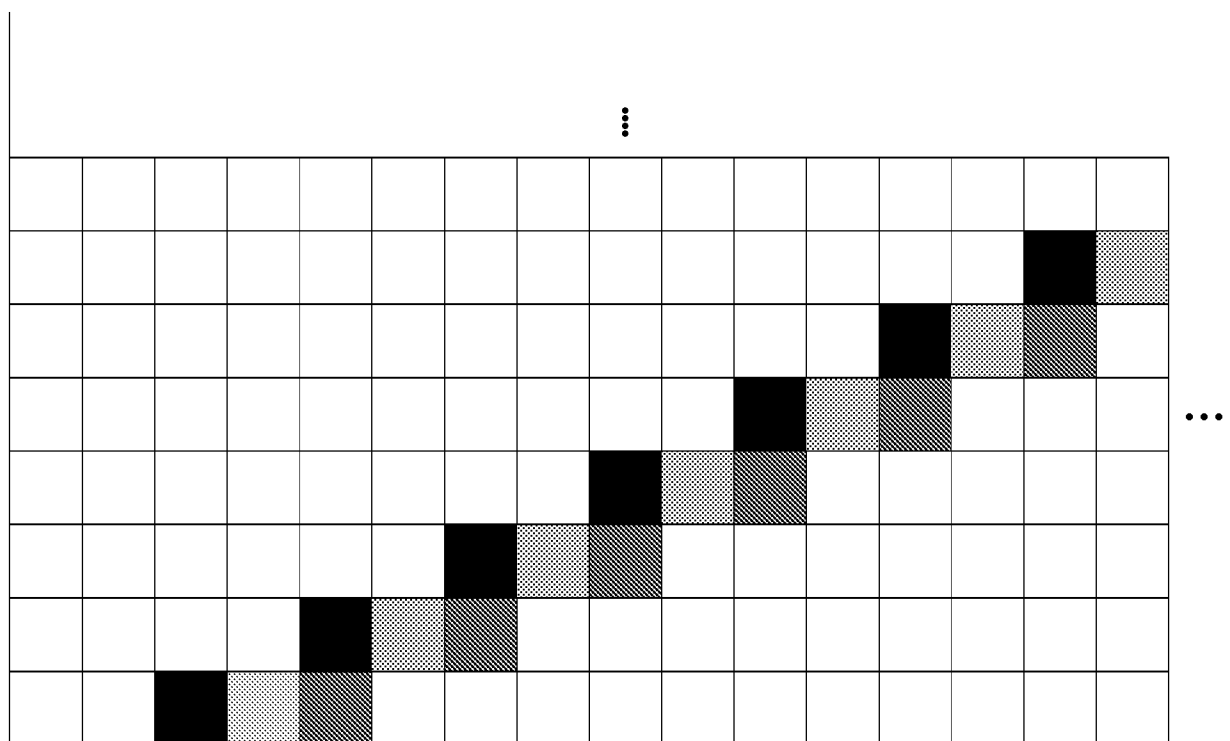


FIG. 9

9/12

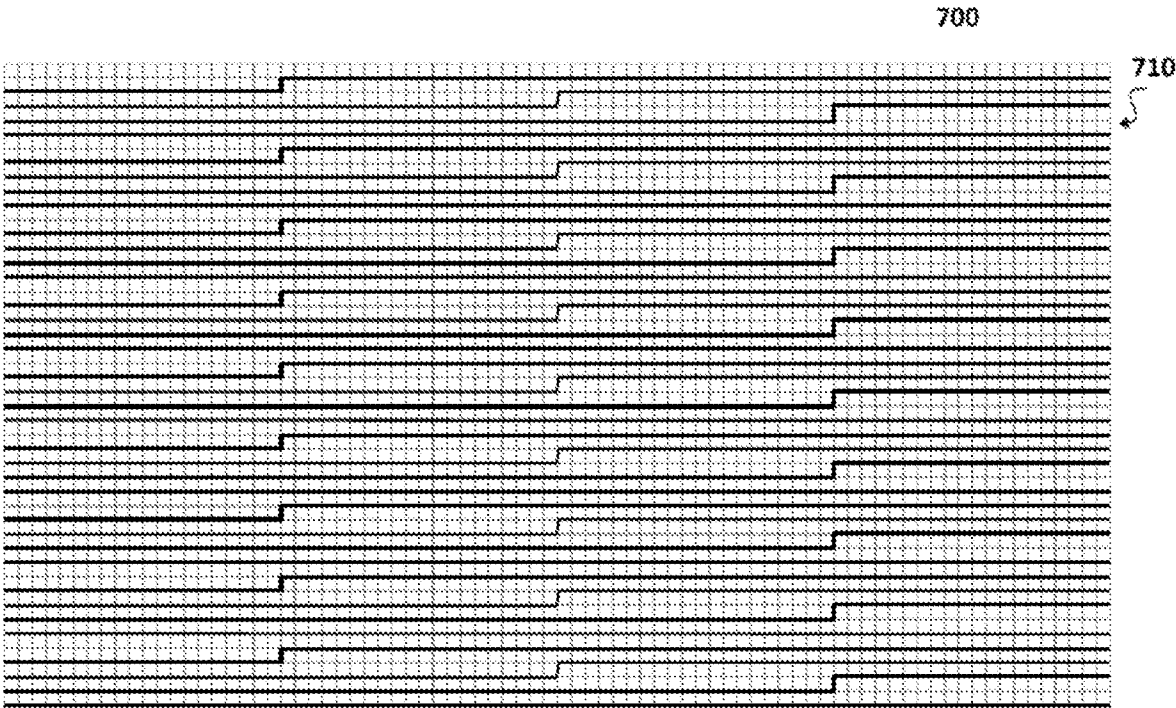


FIG. 10

10/12

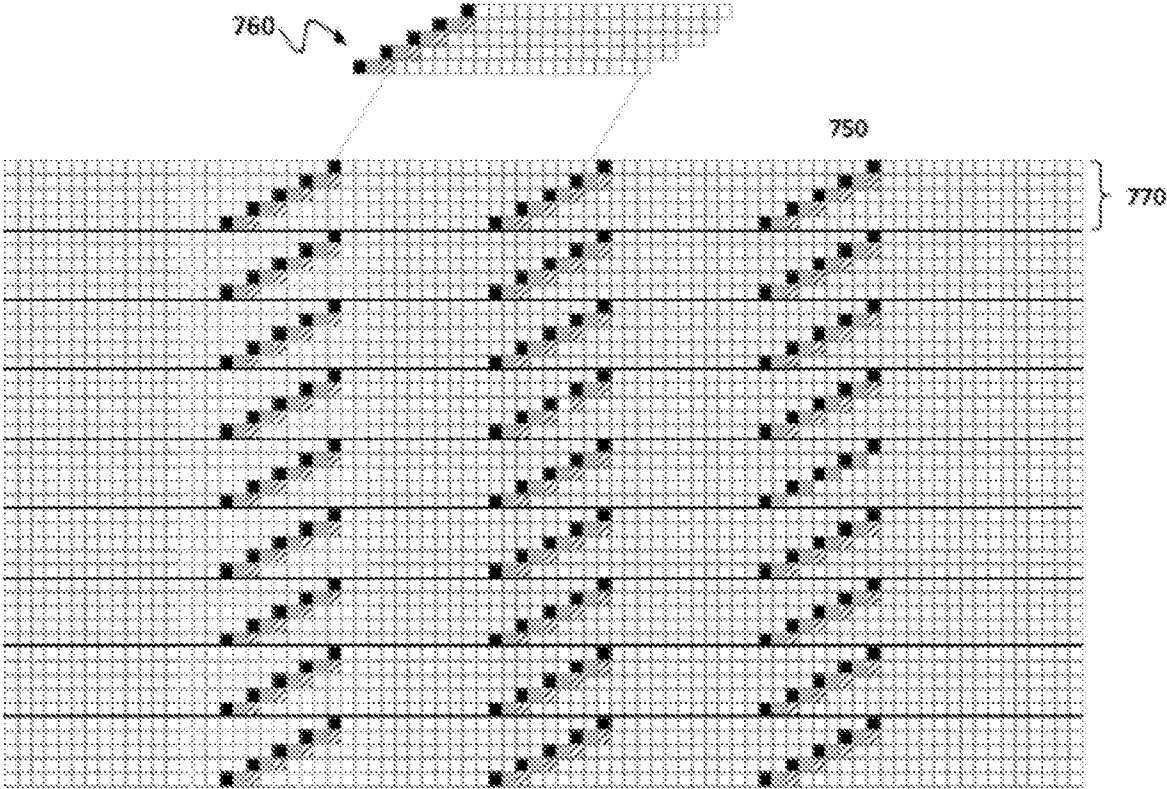


FIG. 11

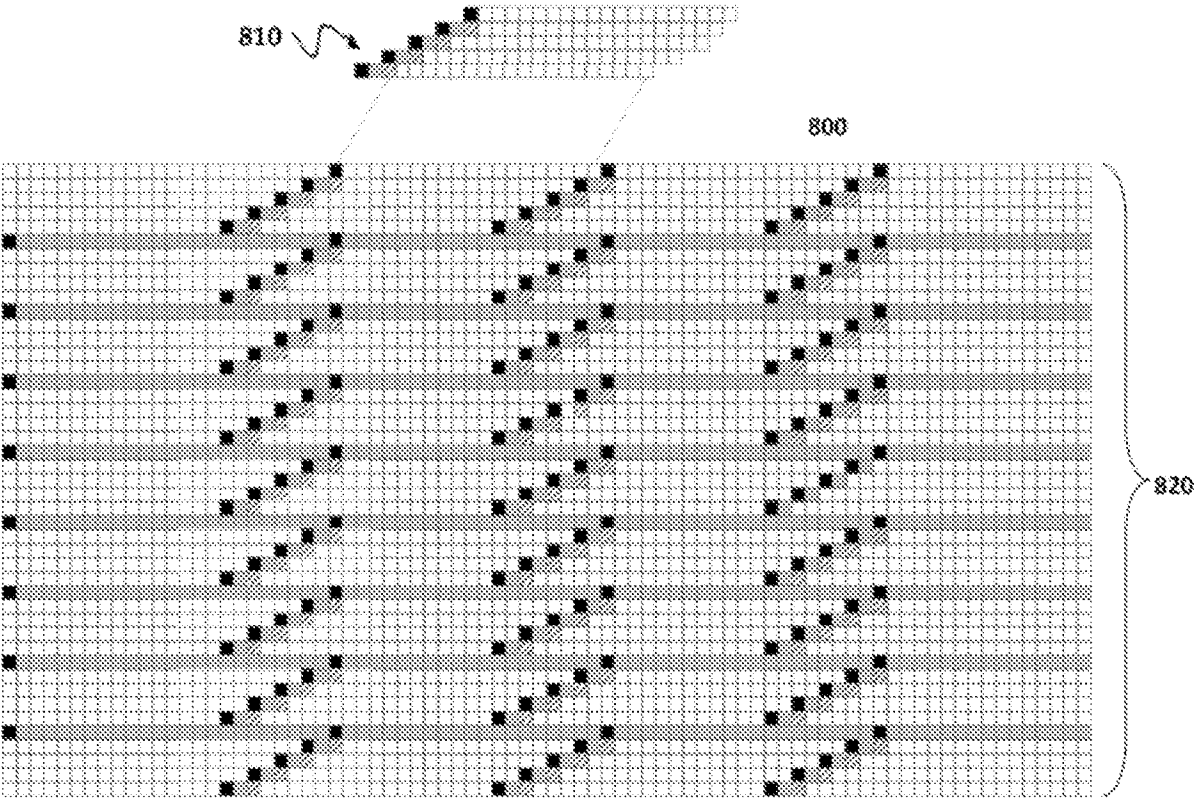


FIG. 12

12/12

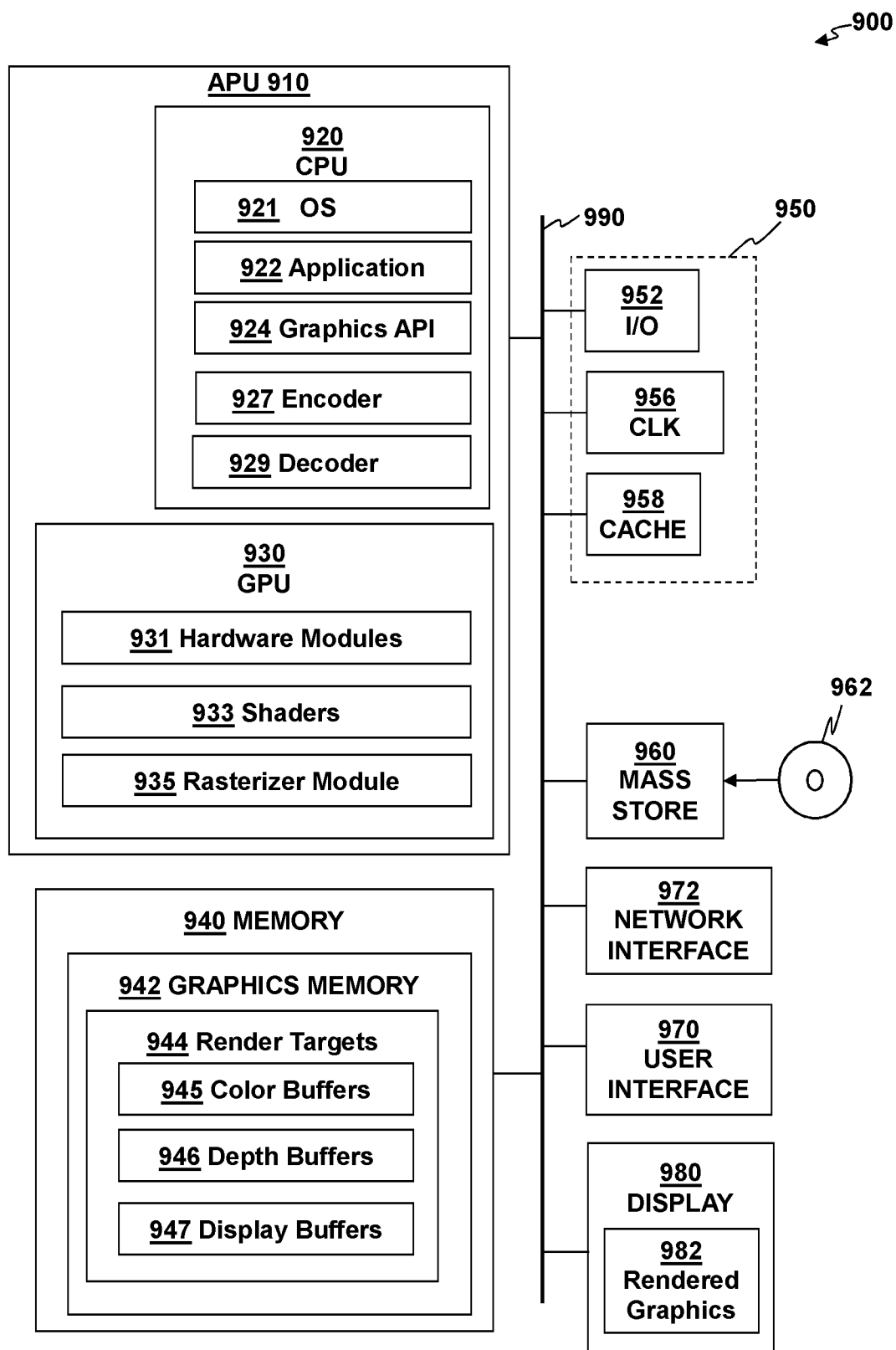


FIG. 13

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US2016/039094

A. CLASSIFICATION OF SUBJECT MATTER

IPC(8) - G06K 9/36, H04N 7/12, H04N 7/36, H04N 11/02 (2016.01)

CPC - H04N 7/26106, H03M 7/4006, H03M 7/42, H04N 19/00733, H04N 19/103, H04N 19/112 (2016.05)

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)

IPC - G06K 9/36, H04N 7/12, H04N 7/36, H04N 11/02

CPC - H04N 7/26106, H03M 7/4006, H03M 7/42, H04N 19/00733, H04N 19/103, H04N 19/112, H04N 19/13, H04N 19/16

H04N 19/176, H04N 19/436, H04N 19/46, H04N 19/51, H04N 19/573, H04N 19/61, H04N 19/70, H04N 19/91

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

USPC - 375/240.02, 375/240.12, 375/240.16, 375/240.24, 375/E07.001, 375/E07.262, 382/236, 382/238 (keyword delimited)

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

Orbit, Google Patents, Google Scholar, Google

Search terms used: encode selected sections of digital picture in first list independently of encoding neighbors

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X — Y	US 2011/0235699 A1 (HUANG et al) 29 September 2011 (29.09.2011), entire document	1, 5, 7, 9-11 — 2-4, 6, 8, 13-15, 17, 18, 20, 21
X — Y	US 2015/0063440 A1 (QUALCOMM INCORPORATED) 05 March 2015 (05.03.2015), entire document	12, 16, 19 — 3, 4, 13-15, 17, 18, 20, 21
Y	US 2011/0260893 A1 (SATO et al) 27 October 2011 (27.10.2011), entire document	2
Y	SULLIVAN et al. "Video compression-from concepts to the H. 264/AVC standard." In: Proceedings of the IEEE. January 2005 (01.2005) Retrieved from <http://ce.sharif.ac.ir/courses/87-88/2/ce342/resources/root/H264%20Standard/pieee_sullivan_wiegand_dec2004.pdf>, entire document	6, 8
A	US 2005/0117646 A1 (JOCH et al) 02 June 2005 (02.06.2005), entire document	1-21
A	US 2014/0064372 A1 (LAROCHE et al) 06 March 2014 (06.03.2014), entire document	1-21
A	US 2005/0053296 A1 (SRINIVASAN et al) 10 March 2005 (10.03.2005), entire document	1-21
A	US 2011/0051811 A1 (WANG et al) 03 March 2011 (03.03.2011), entire document	1-21

☐ 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

22 August 2016

Date of mailing of the international search report

13 SEP 2016

Name and mailing address of the ISA/

Mail Stop PCT, Attn: ISA/US, Commissioner for Patents

P.O. Box 1450, Alexandria, VA 22313-1450

Facsimile No. 571-273-8300

Authorized officer

Blaine R. Copenheaver

PCT Helpdesk: 571-272-4300

PCT OSP: 571-272-7774