Title: DECODING METHOD AND SYSTEM FOR HIGHLY COMPRESSED VIDEO DATA

Abstract: Embodiments of a method and system for motion compensation in decoding video data are described herein. In various embodiments, a high-compression-ratio codec (such as H.264) is part of the encoding scheme for the video data. Embodiments pre-process control maps that were generated from encoded video data, and generating intermediate control maps comprising information regarding decoding the video data. The control maps indicate which one of multiple prediction operations is to be used in performing motion compensation on particular units of data in a frame. In an embodiment, motion compensation is performed on a frame basis such that each of the multiple prediction operations is performed on an entire frame at one time. In other embodiments, processing of different frames is interleaved. Embodiments increase the efficiency of the motion compensation such as to allow decoding of high-compression-ratio encoded video data on personal computers or comparable equipment without special, additional decoding hardware.
ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM),
European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI,
FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL,
PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM,
GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published: without international search report and to be republished
upon receipt of that report
DECODING METHOD AND SYSTEM FOR HIGHLY COMPRESSED VIDEO DATA

RELATED APPLICATIONS
This application is related to the following U.S. Patent applications, each of which is incorporated by reference in its entirety herein:

11/514,780, filed August 31, 2006 titled Decoding system and method for highly compressed video data;
11/514,473, filed August 31, 2006 titled Method and system for inter-prediction in decoding of video data;
11/515,472, filed August 31, 2006 titled Method and system for intra-prediction in decoding of video data;
11/514,801, filed August 31, 2006 titled Method and system for motion compensation in decoding of video data;
11/515,311, filed August 31, 2006 titled Method and system for deblocking in decoding of video data;

TECHNICAL FIELD
The invention is in the field of decoding video data that has been encoded according to a specified encoding format, and more particularly, decoding the video data to optimize use of data processing hardware.

BACKGROUND
Digital video playback capability is increasingly available in all types of hardware platforms, from inexpensive consumer-level computers to super-sophisticated flight simulators. Digital video playback includes displaying video that is accessed from a storage medium or streamed from a real-time source, such as a television signal. As digital video becomes nearly ubiquitous, new techniques to improve the quality and accessibility of the digital video are being developed. For example, in order to store and transmit digital video, it is typically compressed or encoded using a format specified by a standard. Recently H.264, a video compression scheme, or codec, has been adopted by the Motion Pictures Expert Group (MPEG) to be the video compression scheme for the MPEG-4 format for digital media exchange. H.264 is MPEG-4 Part 10. H.264 was developed to address various needs in an
evolving digital media market, such as relative inefficiency of older compression schemes, the availability of greater computational resources today, and the increasing demand for High Definition (HD) video, which requires the ability to store and transmit about six times as much data as required by Standard Definition (SD) video.

H.264 is an example of an encoding scheme developed to have a much higher compression ratio than previously available in order to efficiently store and transmit higher quantities of video data, such as HD video data. For various reasons, the higher compression ratio comes with a significant increase in the computational complexity required to decode the video data for playback. Most existing personal computers (PCs) do not have the computational capability to decode HD video data compressed using high compression ratio schemes such as H.264. Therefore, most PCs cannot playback highly compressed video data stored on high-density media such as optical Blu-ray discs (BD) or HD-DVD discs. Many PCs include dedicated video processing units (VPUs) or graphics processing units (GPUs) that share the decoding tasks with the PC. The GPUs may be add-on units in the form of graphics cards, for example, or integrated GPUs. However, even PCs with dedicated GPUs typically are not capable of BD or HD-DVD playback. Efficient processing of H.264/MPEG-4 is very difficult in a multi-pipeline processor such as a GPU. For example, video frame data is arranged in macro blocks according to the MPEG standard. A macro block to be decoded has dependencies on other macro blocks, as well as intrablock dependencies within the macro block. In addition, edge filtering of the edges between blocks must be completed. This normally results in algorithms that simply complete decoding of each macro block sequentially, which involves several computationally distinct operations involving different hardware passes. This results in failure to exploit the parallelism that is inherent in modern day processors such as multi-pipeline GPUs.

One approach to allowing PCs to playback high-density media is the addition of separate decoding hardware and software. This decoding hardware and software is in addition to any existing graphics card(s) or integrated GPUs on the PC. This approach has various disadvantages. For example, the hardware and software must be provided for each PC which is to have the decoding capability. In addition, the decoding hardware and software decodes the video data without particular consideration for optimizing the graphics processing hardware which will display the decoded data.
It would be desirable to have a solution for digital video data that allows a PC user to playback high-density media such as BD or HD-DVD without the purchase of special add-on cards or other hardware. It would also be desirable to have such a solution that decodes the highly compressed video data for processing so as to optimize the use of the graphics processing hardware, while minimizing the use of the CPU, thus increasing speed and efficiency.

**BRIEF DESCRIPTION OF THE DRAWINGS**

Figure 1 is a block diagram of a system with graphics processing capability according to an embodiment.

Figure 2 is a block diagram of elements of a GPU according to an embodiment.

Figure 3 is a diagram illustrating a data and control flow of a decoding process according to an embodiment.

Figure 4 is another diagram illustrating a data and control flow of a decoding process according to an embodiment.

Figure 5 is a diagram illustrating a data and control flow of an inter-prediction process according to an embodiment.

Figures 6A, 6B, and 6C are diagrams of a macro block divided into different blocks according to an embodiment.

Figure 7 is a block diagram illustrating intra-block dependencies according to an embodiment.

Figure 8 is a diagram illustrating a data and control flow of an intra-prediction process according to an embodiment.

Figure 9 is a block diagram of a frame after inter-prediction and intra-prediction have been performed according to an embodiment.

Figures 10A and 10B are block diagrams of macro blocks illustrating vertical and horizontal deblocking, which are performed on each macro block according to an embodiment.

Figures 11A, 11B, 11C, and 11D show the pels involved in vertical deblocking for each vertical edge in a macro block according to an embodiment.

Figures 12A, 12B, 12C, and 12D show the pels involved in horizontal deblocking for each horizontal edge in a macro block according to an embodiment.
**Figure 13A** is a block diagram of a macro block that shows vertical edges 0-3 according to an embodiment.

**Figure 13B** is a block diagram that shows the conceptual mapping of the shaded data from Figure 13A into a scratch buffer according to an embodiment.

**Figure 14A** is a block diagram that shows multiple macro blocks and their edges according to an embodiment.

**Figure 14B** is a block diagram that shows the mapping of the shaded data from Figure 14A into the scratch buffer according to an embodiment

**Figure 15A** is a block diagram of a macro block that shows horizontal edges 0-3 according to an embodiment

**Figure 15B** is a block diagram that shows the conceptual mapping of the shaded data from Figure 15A into the scratch buffer according to an embodiment

**Figure 16A** is a block diagram that shows multiple macro blocks and their edges according to an embodiment

**Figure 16B** is a block diagram that shows the mapping of the shaded data from Figure 16A into the scratch buffer according to an embodiment.

**Figure 17A** is a block diagram that shows multiple macro blocks and their edges according to an embodiment.

**Figure 17B** is a block diagram that shows the mapping of the shaded data from Figure 17A into the scratch buffer according to an embodiment.

**Figure 18A** is a block diagram that shows multiple macro blocks and their edges according to an embodiment.

**Figure 18B** is a block diagram that shows the mapping of the shaded data from Figure 18A into the scratch buffer according to an embodiment

**Figure 19A** is a block diagram that shows multiple macro blocks and their edges according to an embodiment

**Figure 19B** is a block diagram that shows the mapping of the shaded data from Figure 19A into the scratch buffer according to an embodiment

**Figure 20** is a block diagram of a source buffer at the beginning of a deblocking algorithm iteration according to an embodiment.

**Figure 21** is a block diagram of a target buffer at the beginning of a deblocking algorithm iteration according to an embodiment.

**Figure 22** is a block diagram of the target buffer after the left side filtering according to an embodiment
**Figure 23** is a block diagram of the target buffer after the vertical filtering according to an embodiment.

**Figure 24** is a block diagram of a new target buffer after a copy according to an embodiment.

**Figure 25** is a block diagram of the target buffer after a pass according to an embodiment.

**Figure 26** is a block diagram of the target buffer after a pass according to an embodiment.

**Figure 27** is a block diagram of the target buffer after a copy according to an embodiment.

The drawings represent aspects of various embodiments for the purpose of disclosing the invention as claimed, but are not intended to be limiting in any way.

**DETAILED DESCRIPTION**

Embodiments of a method and system for layered decoding of video data encoded according to a standard that includes a high-compression ratio compression scheme are described herein. The term "layer" as used herein indicates one of several distinct data processing operations performed on a frame of encoded video data in order to decode the frame. The distinct data processing operations include, but are not limited to, motion compensation and deblocking. In video data compression, motion compensation typically refers to accounting for the difference between consecutive frames in terms of where each section of the former frame has moved to. In an embodiment, motion compensation is performed using inter-prediction and/or intra-prediction, depending on the encoding of the video data.

Prior decoding methods performed all of the distinct data processing operations on a unit of data within the frame before moving to a next unit of data within a frame. In contrast, embodiments of the invention perform a layer of processing on an entire frame at one time, and then perform a next layer of processing. In other embodiment, multiple frames are processed in parallel using the same algorithms described below. The encoded data is pre-processed in order to allow layered decoding without errors, such as errors that might result from processing interdependent data in an incorrect order. The pre-processing prepares various sets of encoded data to be operated on in parallel by different processing
pipelines, thus optimizing the use of the available graphics processing hardware and
minimizing the use of the CPU.

Figure 1 is a block diagram of a system 100 with graphics processing
capability according to an embodiment. The system 100 includes a video data source
112. The video data source 112 may be a storage medium such as a Blu-ray disc or
an HD-DVD disc. The video data source may also be a television signal, or any other
source of video data that is encoded according to a widely recognized standard, such
as one of the MPEG standards. Embodiments of the invention will be described with
reference to the H.264 compression scheme, which is used in the MPEG-4 standard.
Embodiments provide particular performance benefits for decoding H.264 data, but
the invention is not so limited. In general, the particular examples given are for
thorough illustration and disclosure of the embodiments, but no aspects of the
examples are intended to limit the scope of the invention as defined by the claims.

System 100 further includes a central processing unit (CPU)-based processor
108 that receives compressed, or encoded, video data 109 from the video data source
112. The CPU-based processor 108, in accordance with the standard governing the
encoding of the data 109, processes the data 109 and generates control maps 106 in a
known manner. The control maps 106 include data and control information formatted
in such a way as to be meaningful to video processing software and hardware that
further processes the control maps 106 to generate a picture to be displayed on a
screen. In an embodiment, the system 100 includes a graphics processing unit (GPU)
102 that receives the control maps 106. The GPU 102 may be integral to the system
100. For example, the GPU 102 may be part of a chipset made for inclusion in a
personal computer (PC) along with the CPU-based processor 108. Alternatively, the
GPU 102 may be a component that is added to the system 100 as a graphics card or
video card, for example. In embodiments described herein, the GPU 102 is designed
with multiple processing cores, also referred to herein as multiple processing pipelines
or multiple pipes. In an embodiment, the multiple pipelines each contain similar
hardware and can all be run simultaneously on different sets of data to increase
performance. In an embodiment, the GPU 102 can be classed as a single instruction
multiple data (SIMD) architecture, but embodiments are not so limited.

The GPU 102 includes a layered decoder 104, which will be described in
greater detail below. In an embodiment, the layered decoder 104 interprets the
control maps 106 and pre-processes the data and control information so that
processing hardware of the GPU 102 can optimally perform parallel processing of the data. The GPU 102 thus performs hardware-accelerated video decoding. The GPU 102 processes the encoded video data and generates display data 115 for display on a display 114. The display data 115 is also referred to herein as frame data or decoded frames. The display 114 can be any type of display appropriate to a particular system 100, including a computer monitor, a television screen, etc.

In order to facilitate describing the embodiments, an overview of the type of video data that will be referred to in the description now follows. A SIMD architecture is most effective when it conducts multiple, massively parallel computations along substantially the same control flow path. In the examples described herein, embodiments of the layered decoder 104 include an H.264 decoder running GPU hardware to minimize the flow control deviation in each shader thread. A shader as referred to herein is a software program specifically for rendering graphics data or video data as known in the art. A rendering task may use several different shaders.

The following is a brief explanation of some of the terminology used in this description.

A luma or chroma 8-bit value is called a pel. All luma pels in a frame are named in the Y plane. The Y plane has a resolution of the picture measured in pels. For example, if the picture resolution is said to be 720x480, the Y plane has 720x480 pels. Chroma pels are divided into two planes: a U plane and a V plane. For purposes of the examples used to describe the embodiments herein, a so-called 420 format is used. The 420 format uses U and V planes having the same resolution, which is half of the width and height of the picture. In a 720x480 example, the U and V resolution is 360x240 measured in pels.

Hardware pixels are pixels as they are viewed by the GPU on the read from memory and the write to the memory. In most cases this is a 4-channel, 8-bit per channel pixel commonly known as RGBA or ARGB.

As used herein, "pixel" also denotes a 4x4 pel block selected as a unit of computation. It means that as far as the scan converter is concerned this is the pixel, causing the pixel shader to be invoked per each 4x4 block. In an embodiment, to accommodate this view, the resolution of the target surface presented to the hardware is defined as one quarter of the width and of the height of the original picture.
resolution measured in pels. For example, returning to the 720x480 picture example, the resolution of the target is 180x120.

The block of 16x16 pels, also referred to as a macro block, is the maximal semantically unified chunk of video content, as defined by MPEG standards. A block of 4x4 pels is the minimal semantically unified chunk of the video content.

There are 3 different physical target picture or target frame layouts employed depending on the type of the picture being decoded. The target frame layouts are illustrated in Tables 1-3.

Let PicWidth be the width of the picture in pels (which is the same as bytes) and PicHeight be the height of the picture in scan lines (for example, 720x480 in the previous example. Table 1 shows the physical layout based on the picture type.

Table 1

<table>
<thead>
<tr>
<th>Frame/AFF</th>
<th>Field</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>{0,0}, {PicWidth-1,PicHeight-1}</td>
<td>{0,0}, {PicWidth-1,PicHeight/2-1}</td>
</tr>
<tr>
<td>U</td>
<td>{0,PicHeight}, {PicWidth/2-1,3*PicHeight/2-1}</td>
<td>{0,PicHeight}, {PicWidth/2-1,5*PicHeight/4-1}</td>
</tr>
<tr>
<td>V</td>
<td>{PicWidth/2,PicHeight}, {PicWidth-1,3*PicHeight/2-1}</td>
<td>{PicWidth/2,PicHeight}, {PicWidth-1,5*PicHeight/4-1}</td>
</tr>
</tbody>
</table>
Following Tables 2 and 3 are visual representations of Table 1 for a frame/AFF picture and for a field picture, respectively.

**Table 2**

<table>
<thead>
<tr>
<th>Frame/AFF picture</th>
<th>Y plane</th>
</tr>
</thead>
<tbody>
<tr>
<td>U plane</td>
<td>V plane</td>
</tr>
</tbody>
</table>

**Table 3**

<table>
<thead>
<tr>
<th>Field picture</th>
<th>Y plane even</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y plane odd</td>
<td></td>
</tr>
<tr>
<td>U plane even</td>
<td>V plane even</td>
</tr>
<tr>
<td>U plane odd</td>
<td>V plane odd</td>
</tr>
</tbody>
</table>

The field type picture keeps even and odd fields separately until a "interleaving" pass. The AFF type picture keeps field macro blocks as two complimentary pairs until the last "interleaving" pass. The interleaving pass interleaves even and odd scan lines and builds one progressive frame.
Embodiments described herein include a hardware decoding implementation of the H.264 video standard. H.264 decoding contains three major parts: inter-prediction; intra-prediction; and deblocking filtering. In various embodiments, inter-prediction and intra-prediction are also referred to as motion compensation because of the effect of performing inter-prediction and intra-prediction.

According to embodiments a decoding algorithm consists of three "logical" passes. Each logical pass adds another layer of data onto the same output picture or frame. The first "logical" pass is the inter-prediction pass with added inverted transformed coefficients. The first pass produces a partially decoded frame. The frame includes macro blocks designated by the encoding process to be decoded using either inter-prediction or intra-prediction. Because only the inter-prediction macro blocks are decoded in the first pass, there will be "holes" or "garbage" data in place of intra-prediction macro blocks.

A second "logical" pass touches only intra-prediction macro blocks left after the first pass is complete. The second pass computes the intra-prediction with added inverted transformed coefficients.

A third pass is a deblocking filtering pass, which includes a deblock control map generation pass. The third pass updates pels of the same picture along the sub-block (e.g., 4x4 pels) edges.

The entire decoding algorithm as further described herein does not require intervention by the host processor or CPU. Each logical pass may include many physical hardware passes. In an embodiment, all of the passes are pre-programmed by a video driver, and the GPU hardware moves from one pass to another autonomously.

Figure 2 is a block diagram of elements of a GPU 202 according to an embodiment. The GPU 202 receives control maps 206 from a source such as a host processor or host CPU. The GPU 202 includes a video driver 222 which, in an embodiment, includes a layered decoder 204. The GPU 202 also includes processing pipelines 220A, 220B, 220C, and 220D. In various embodiments, there could be less than four or more than four pipelines 220. In other embodiments, more than one GPU 202 may be combined to share processing tasks. The number of pipelines is not intended to be limiting, but is used in this description as a convenient number for illustrating embodiments of the invention. In many embodiments, there are
significantly more than four pipelines. As the number of pipelines is increased, the speed and efficiency of the GPU is increased.

An advantage of the embodiments described is the flexibility and ease of use provided by the layered decoder 204 as part of the driver 222. The driver 222, in various embodiments, is software that can be downloaded by a user of an existing GPU to extend new layered decoding capability to the existing GPU. The same driver can be appropriate for all existing GPUs with similar architectures. Multiple drivers can be designed and made available for different architectures. One common aspect of drivers including layered decoders described herein is that they immediately allow efficient decoding of video data encoded using H.264 and similar formats by maximizing the use of available graphics processing pipelines on an existing GPU.

The GPU 202 further includes a Z-buffer 216 and a reference buffer 218. As further described below, Z buffer is used as control information, for example to decide which macro blocks are processed and which are not in any layer. The reference buffer 218 is used to store a number of decoded frames in a known manner. Previously decoded frames are used in the decoding algorithm, for example to predict what a next or subsequent frame might look like.

**Figure 3** is a diagram illustrating a flow of data and control in layered decoding according to an embodiment. Control maps 306 are generated by a host processor such as a CPU, as previously described. The control maps 306 are generated according to the applicable standard, for example MPEG-4. The control maps 306 are generated on a per-frame basis. A control map 306 is received by the GPU (as shown in Figures 1 and 7). The control maps 306 include various information used by the GPU to direct the graphics processing according to the applicable standard. For example, as previously described, the video frame is divided into macro blocks of certain defined sizes. Each macro block may be encoded such that either inter-prediction or intra-prediction must be used to decode it. The decision to encode particular macro blocks in particular ways is made by the encoder. One piece of information conveyed by the control maps 306 is which decoding method (e.g., inter-prediction or intra-prediction) should be applied to each macro block.

Because the encoding scheme is a compression of data, one of the aspects of the overall scheme is a comparison of one frame to the next in time to determine what video data does not change, and what video data changes, and by how much. Video data that does not change does not need to be explicitly expressed or transmitted, thus
allowing compression. The process of decoding, or decompression, according to the MPEG standards, involves reading information in the control maps 306 including this change information per unit of video data in a frame, and from this information, assembling the frame. For example, consider a macro block whose intensity value has changed from one frame to another. During inter-prediction, the decoder reads a residual from the control maps 306. The residual is an intensity value expressed as a number. The residual represents a change in intensity from one frame to the next for a unit of video data.

The decoder must then determine what the previous intensity value was and add the residual to the previous value. The control maps 306 also store a reference index. The reference index indicates which previously decoded frame of up to sixteen previously decoded frames should be accessed to retrieve the relevant, previous reference data. The control maps also store a motion vector that indicates where in the selected reference frame the relevant reference data is located. In an embodiment, the motion vector refers to a block of 4x4 pels, but embodiments are not so limited.

The GPU performs preprocessing on the control map 306, including setup passes 308, to generate intermediate control maps 307. The setup passes 308 include sorting surfaces for performing inter-prediction for the entire frame, intra-prediction for the entire frame, and deblocking for the entire frame, as further described below. The setup passes 308 also include intermediate control map generation for deblocking passes according to an embodiment. The setup passes 308 involve running "pre-shaders" that can be referred to as software programs of relatively small size (compared to the usual rendering shaders) to read the control map 306 without incurring the performance penalty for running the usual rendering shaders.

In general, the intermediate control maps 307 are the result of interpretation and reformulation of control map 306 data and control information so as to tailor the data and control information to run in parallel on the particular GPU hardware in an optimized way.

In yet other embodiments, all the control maps are generated by the GPU. The initial control maps are CPU-friendly and data is arranged per macro block. Another set of control maps can be generated from the initial control maps using the GPU, where data is arranged per frame (for example, one map for motion vectors, one map for residual).
After setup passes 308 generate intermediate control maps 307, shaders are run on the GPU hardware for inter-prediction passes 310. In some cases, inter-prediction passes 310 may not be available because the frame was encoded using intra-prediction only. It is also possible for a frame to be encoded using only inter-prediction. It is also possible for deblocking to be omitted.

The inter-prediction passes are guided by the information in the control maps 306 and the intermediate control maps 307. Intermediate control maps 307 include a map of which macro blocks are inter-prediction macro blocks and which macro blocks are intra-prediction macro blocks. Inter-prediction passes 310 read this “inter-intra” information and process only the macro blocks marked as inter-prediction macro blocks. The intermediate control maps 307 also indicate which macro blocks or portions of macro blocks may be processed in parallel such that use of the GPU hardware is optimized. In our example embodiment there are four pipelines which process data simultaneously in inter-prediction passes 310 until inter-prediction has been completed on the entire frame. In other embodiments, the solution described here can be scaled with the hardware such that more pipelines allow simultaneous processing of more data.

When the inter-prediction passes 310 are complete, and there are intra-predicted macro blocks, there is a partially decoded frame 312. AU of the inter-prediction is complete for the partially decoded frame 312, and there are "holes" for the intra-prediction macro blocks. In some cases, the frame may be encoded using only inter-prediction, in which case the frame has no "holes" after inter-prediction.

Intra-prediction passes 314 use the control maps 306 and the intermediate control maps 307 to perform intra-prediction on all of the intra-prediction macro blocks of the frame. The intermediate control maps 307 indicate which macro blocks are intra-prediction macro blocks. Intra-prediction involves prediction of how a unit of data will look based on neighboring units of data within a frame. This is in contrast to inter-prediction, which is based on differences between frames. In order to perform intra-prediction on a frame, units of data must be processed in an order that does not improperly overwrite data.

When the intra-prediction passes 314 are complete, there is a partially decoded frame 316. All of the inter-prediction and intra-prediction operations are complete for the partially decoded frame 316, but deblocking is not yet performed. Decoding on a macro block level causes a potentially visible transition on the edges between macro
blocks. Deblocking is a filtering operation that smoothes these transitions. In an embodiment, the intermediate control maps 307 include a deblocking map (if available) that indicates an order of edge processing and also indicates filtering parameter. No deblocking map is available if deblocking is not required. In deblocking, the data from adjacent macro block edges is combined and rewritten so that the visible transition is minimized. In an embodiment, the data to be operated on is written out to scratch buffers 322 for the purpose of rearranging the data to be optimally processed in parallel on the hardware, but embodiments are not so limited.

After the deblocking passes 318, a completely decoded frame 320 is stored in the reference buffer (reference buffer 218 of Figure 2, for example). This is the reference buffer accessed by the inter-prediction passes 310, as shown by arrow 330.

**Figure 4** is another diagram illustrating a flow 400 of data and control in video data decoding according to an embodiment. Figure 4 is another perspective of the operation illustrated in Figure 3 with more detail. Control maps 406 are received by the GPU. In order to generate an intermediate control map that indicates which macro blocks are for inter-prediction, a comparison value in the Z-buffer is set to "inter" at 408. The comparison value can be a single bit that is set to "1" or "0", but embodiments are not so limited. With the comparison value set to "inter", a small shader, or "pre-shader" 410 is run on the control maps 406 to create the Z-buffer 412 and intermediate control maps 413. The Z-buffer includes information that tells an inter-prediction shader 414 which macro blocks are to be inter-predicted and which are not. In an embodiment this information is determined by Z-testing, but embodiments are not so limited. Macro blocks that are not indicated as inter-prediction macro blocks will not be processed by the inter-prediction shader 414, but will be skipped or discarded. The inter-prediction shader 414 is run on the data using control information from control maps 406 and an intermediate control map 413 to produce a partially decoded frame 416 in which all of the inter-prediction macro blocks are decoded, and all of the remaining macro blocks are not decoded. In another implementation, the Z buffer testing of whether a macro block is an inter-prediction macro block or an intra-prediction macro block is performed within the inter prediction shader 414.

The value set at 408 is then reset at 418 to indicate intra-prediction. In another embodiment, the value is not reset, but rather another buffer is used. A pre-shader 420 creates a Z-buffer 415 and intermediate control maps 422. The Z-buffer includes
information that tells an intra-prediction shader 424 which macro blocks are to be intra-predicted and which are not. In an embodiment this information is determined by Z-testing, but embodiments are not so limited. Macro blocks that are not indicated as intra-prediction macro blocks will not be processed by the intra-prediction shader 424, but will be skipped or discarded. The inter-prediction shader 424 is run on the data using control information from control maps 406 and an intermediate control map 422 to produce a frame 426 in which all of the inter-prediction macro blocks are decoded and all of the intra-prediction macro blocks are decoded. This is the frame that is processed in the deblocking operation.

Inter-prediction

As previously discussed, inter-prediction is a way to use pels from reference pictures or frames (future (forward) or past (backward)) to predict the pels of the current frame. Figure 5 is a diagram illustrating a data and control flow of an inter-prediction process 500 for a frame according to an embodiment. In an embodiment, the geometrical mesh for each inter-prediction pass consists of a grid of 4x4 rectangles in the Y part of the physical layout and 2x2 rectangles in the UV part (16x16 or 8x8 pels, where 16x16 pels is a macro block). A shader (in an embodiment, a vertex shader) parses the control maps for each macro block's control information and broadcasts the preprocessed control information to each pixel (in this case, a pixel is a 4x4-block). The control information includes an 8-bit macro block header, multiple IT coefficients and their offsets, 16 pairs of motion vectors and 8 reference frame selectors. Z-testing as previously described indicates whether the macro block is not an inter-prediction block, in which case, its pixels will be "killed" or skipped from "rendering".

At 504, a particular reference frame among various reference frames in the reference buffer is selected using the control information. Then, at 506, the reference pels within the reference frame are found. In an embodiment, finding the correct position of the reference pels inside the reference frame includes computing the coordinates for each 4x4 block. The input to the computation is the top-left address of the target block in pels, and the delta obtained from the proper control map. The target block is the destination block, or the block in the frame that is being decoded.

As an example of finding reference pels, let MvDx, MvDy be the delta obtained from the control map. MvDx, MvDy are the x, y deltas computed in the appropriate coordinate system. This is true for a frame picture and frame macro block
of an AFF picture in frame coordinates, and for a field picture and field macro block of an AFF picture in the field coordinate system of proper polarity. In an embodiment, the delta is the delta between the X,Y coordinates of the target block and the X,Y coordinates of the source (reference) block with 4-bit fractional precision.

When the reference pels are found, they are combined at 508 with the residual data (also referred to as "the residual") that is included in the control maps. The result of the combination is written to the destination in the partially decoded frame at 512. The process 500 is a parallel process and all blocks are submitted/executed in parallel. At the completion of the process, the frame data is ready for intra-prediction. In an embodiment, 4x4 blocks are processed in parallel as described in the process 500, but this is just an example. Other units of data could be treated in a similar way.

Intra-prediction

As previously discussed, intra-prediction is a way to use pels from other macro blocks or portions of macro blocks within a pictures or frame to predict the pels of the current macro block or portion of a macro block. Figures 6A, 6B, and 6C are diagrams of a macro block divided into different blocks according to an embodiment. Figure 6A is a diagram of a macro block that includes 16x16 pels. Figure 6B is diagram of 8x8 blocks in a macro block. Figure 6C is a diagram of 4x4 blocks in a macro block. Various intra-prediction cases exist depending on the encoding performed. For example, macro blocks in a frame may be divided into sub-blocks of the same size. Each sub-block may have from 8 cases to 14 cases, or shader branches. The frame configuration is known before decoding from the control maps.

In an embodiment, a shader parses the control maps to obtain control information for a macro block, and broadcasts the preprocessed control information to each pixel (in this case, a pixel is a 4x4-block). The information includes an 8-bit macro block header, a number of IT coefficients and their offsets, availability of neighboring blocks and their types, and for 16x16 and 8x8 blocks, prediction values and prediction modes. Z-testing as previously described indicates whether the macro block is not an intra-prediction block, in which case, its pixels will be "killed" or skipped from "rendering".

Dependencies exist between blocks because data from an encoded (not yet decoded) block should not be used to intra-predict a block. Figure 7 is a block diagram that illustrates these potential intra-block dependencies. Sub-block 702
depends on its neighboring sub-blocks 704 (left), 706 (up-left), 708 (up), and 710 (up-right).

To avoid interdependencies inside the macro block the 16 pixels inside a 4x4 rectangle (Y plane) are rendered in a pass number indicated inside the cell. The intra-prediction for a UV macro block and a 16x16 macro block are processed in one pass. Intra-prediction for an 8x8 macro block is computed in 4 passes; each pass computes the intra-prediction for one 8x8 block from left to right and from top to bottom. Table 4 illustrates an example of ordering in a 4x4 case.

Table 4

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 5

To avoid interdependencies between the macro blocks the primitives (blocks of 4x4 pels) rendered in the same pass are organized into a list in a diagonal fashion.

Each cell below in Table 5 is a 4x4 (pixel) rectangle. The number inside the cell connects rectangles belonging to the same lists. Table 5 is an example for 16*8 x 16*8 in the Y plane:
The diagonal arrangement keeps the following relation invariant separately for Y, U and V parts of the target surface:

Frame/Field picture:

if \( k \) is the pass number, \( k > 0 \) \&\& \( k < \text{DiagonalLength} - 1 \), \( \text{MbMU}[2] \) are coordinates of the macro block in the list, then \( \text{MbMU}[I] + \text{MbMU}[O] / 2 + 1 = k \).

An AFF picture makes the process slightly more complex.

The same example as above with an AFF picture is illustrated in Table 6.

Table 6

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
</tbody>
</table>
Inside all of the macro blocks, the pixel tendering sequence stays the same as described above.

There are three types of intra predicted blocks from the perspective of the shader: 16x16 blocks, 8x8 blocks and 4x4 blocks. The driver provides an availability mask for each type of block. The mask indicates which neighbor (upper, upper-right, upper-left or left is available). How the mask is used depends on the block. For some blocks not all masks are needed. For some blocks, instead of the upper-right masks, two left masks are used, etc. If the neighboring macro block is available, the pixels from it are used for the target block prediction according to the prediction mode provided to the shader by the driver.

There are two types of neighbors: upper (upper-right, upper, upper-left) and left

The following describes computation of neighboring pel coordinates for different temporal types of macro blocks of different picture types according to an embodiment.

**EvenMbXPU** is a x coordinate of the complimentary pair of macro block

**EvenMbYPU** is a y coordinate of the complimentary pair of macro block

**YPU** is y coordinate of the current scan line.

**MbXPU** is a x coordinate of the macro block containing the YPU scan line

**MbYPU** is a y coordinate of the macro block containing the YPU scan line
MbYMU is a y coordinate of the same macro block in macro block units
MbYSzPU is a size of the macro block in Y direction.

Frame/Field picture:
Function to compute x,y coordinates of pels in the neighboring macro block to the left:

\[
X_{\text{NeighborPU}} = \text{MbXPU} - 1 \\
Y_{\text{NeighborPU}} = \text{YPU}
\]

Function to compute x,y coordinates of pels in the neighboring macro block to the up:

\[
X_{\text{NeighborPU}} = \text{MbXPU} \\
Y_{\text{NeighborPU}} = \text{MbYPU} - 1;
\]

AFF picture:
Function to compute x,y coordinates of pels in the neighboring macro block to the left:

\[
\text{EvenMbYPU} = (\text{MbYMU} / 2) * 2 \\
X_{\text{NeighborPU}} = \text{MbXPU} - 1 \\
\text{Frame->Frame·} \\
\text{Field ->Field:} \\
\text{Y_{NeighborPU}} = \text{YPU} \\
\text{break;}
\]

Field->Field:

// Interleave scan lines from even and odd neighboring field macro block
\[
\text{YIsOdd} = \text{YPU}\%2 \\
\text{Y_{NeighborPU}} = \text{EvenMbYPU} + (\text{YPU} - \text{EvenMbYPU})/2 + \text{YIsOdd} * \\
\text{MbYSzPU}
\]

break;

Field->Frame:

// Take only even or odd scan lines from the neighboring pair of frame macro blocks.
\[
\text{MblsOdd} = \text{MbYMU} \% 2 \\
\text{Y_{NeighborPU}} = \text{EvenMbYPU} + (\text{YPU} - \text{MbYPU})*2 + \text{MblsOdd}
\]
Function to compute x,y coordinates of pels in the neighboring macro block to the up:

\[ \text{MblsOdd} = \text{MbYM} \mod 2 \]
\[ \text{XNeighbrPU} = \text{MbXPU} \]
Frame -> Frame:
Frame -> Field:
\[ \text{YNeighbrPU} = \text{MbYPU} \cdot (1 - \text{MbYSzPU}) \]
break;
Field -> Field.
\[ \text{MblsOdd} = 1; // it allows always to elevate into the macro block of the same polarity. \]
Field -> Frame:
\[ \text{YNeighbrPU} = \text{MbYPU} - \text{MbYSzPU} \cdot \text{MblsOdd} + \text{MblsOdd} - 2 , \]
break;

**Figure 8** is a diagram illustrating a data and control flow 800 of an intra-prediction process according to an embodiment. At 802, the layered decoder parses the control map macro block header to determine types of subblocks within a macro block. The subblocks identified to be rendered in the same physical pass are assigned the same number "X" at 804. To avoid interdependencies between macro blocks, primitives to be rendered in the same pass are organized into lists in a diagonal fashion at 805. A shader is run on the subblocks with the same number "X" at 806. The subblocks are processed on the hardware in parallel using the same shader, and the only limitation on the amount of data processed at one time is the amount of available hardware.

At 808, it is determined whether number "X" is the last number among the numbers designating subblocks yet to be processed. IfX" is not the last number, the process returns to 806 to run the shader on subblocks with a new number "X". If "X" is the last number, then the frame is ready for the deblocking operation.

**Deblocking Filtering**

After inter-prediction and intra-prediction are completed for the entire frame, the frame is an image without any "holes" or "garbage". The edges between and inside macro blocks are filtered with a deblocking filter to ease the transition that results from
decoding on a macro block level. Figure 9 is a block diagram of a frame 902 after inter-prediction and intra-prediction have been performed. Figure 9 illustrates the deblocking interdependency among macro blocks. Some of the macro blocks in frame 902 are shown and numbered. Each macro block depends on its neighboring left and top macro blocks, meaning these left and top neighbors must be deblocked first. For example, macro block 0 has no dependencies on other macro blocks. Macro blocks 1 each depend on macro block 0, and so on. Each similarly numbered macro block has similar interdependencies. Embodiments of the invention exploit this arrangement by recognizing that all of the similar macro blocks can be rendered in parallel. In an embodiment, each diagonal strip is rendered in a separate pass. The deblocking operation moves through the frame 902 to the right and down as shown by the arrows in Figure 9.

Figures 10A and 10B are block diagrams of macro blocks illustrating vertical and horizontal deblocking, which are performed on each macro block. Figure 10A is a block diagram of a macro block 1000 that shows how vertical deblocking is arranged. Macro block 1000 is 16x16 pels, as previously defined. This includes 16x4 pixels as pixels are defined in an embodiment. The numbered dashed lines 0, 1, 2, and 3 designate vertical edges to be deblocked. In other embodiment there may be more or less pels per pixel, for example depending on a GPU architecture.

Figure 10B is a block diagram of the macro block 1000 that shows how horizontal deblocking is arranged. The numbered dashed lines 0, 1, 2, and 3 designate horizontal edges to be deblocked.

Figures HA, HB, HC, and HD show the pels involved in vertical deblocking for each vertical edge in the macro block 1000. In Figure 11A, the shaded pels, including pels from a previous (left neighboring) macro block are used in the deblocking operation for edge 0.

In Figure 11b, the shaded pels on either side of edge 1 are used in a vertical deblocking operation for edge 1.

In Figure 11C, the shaded pels on either side of edge 2 are used in a vertical deblocking operation for edge 2.

In Figure 11D, the shaded pels on either side of edge 3 are used in a vertical deblocking operation for edge 3.
Figures 12A, 12B, 12C, and 12D show the pels involved in horizontal deblocking for each horizontal edge in the macro block 1000. In Figure 12A, the shaded pels, including pels from a previous (top neighboring) macro block are used in the deblocking operation for edge 0.

In Figure 12b, the shaded pels on either side of edge 1 are used in a horizontal deblocking operation for edge 1.

In Figure 12C, the shaded pels on either side of edge 2 are used in a horizontal deblocking operation for edge 2.

In Figure 12D, the shaded pels on either side of edge 3 are used in a horizontal deblocking operation for edge 3.

In an embodiment, the pels to be processed in the deblocking algorithm are copied to a scratch buffer (for example, see Figure 3) in order to optimally arrange the pel data to be processed for a particular graphics processing, or video processing architecture. A unit of data on which the hardware operates is referred to as a "quad". In an embodiment, a quad is 2x2 pixels, where a pixel is meant as a "hardware pixels". A hardware pixel can be 2x2 of 4x4 pels, 8x8 pels, or 2x2 of ARGB pixels, or others arrangements. In an embodiment, the data to be processed in horizontal deblocking and vertical deblocking is first remapped onto a quad structure in the scratch buffer. The deblocking processing is performed and the result is written to the scratch buffer, then back to the frame in the appropriate location. In the example architecture, the pels are grouped to exercise all of the available hardware. The pels to be processed together may come from anywhere in the frame as long as the macro blocks from which they come are all of the same type. Having the same type means having the same macro block dependencies. The use of a quad as a unit of data to be processed and the processing of four quads at one time are just one example of an implementation. The same principles applied in rearranging the pel data for processing can be applied to any different graphics processing architecture.

In an embodiment, deblocking is performed for each macro block starting with a vertical pass (vertical edge 0, vertical edge 1, vertical edge 2, vertical edge 3) and then a horizontal pass (horizontal edge 0, horizontal edge 1, horizontal edge 2, horizontal edge 3). The parallelism inherent in the hardware design is exploited by processing macro blocks that have no dependencies (also referred to as being independent) together. According to various embodiments, any number of independent macro blocks at may be processed at the same time, limited only by the hardware.
Figures 13-19 are block diagrams that illustrate mapping to the scratch buffer according to an embodiment. These diagrams are an example of mapping to accommodate a particular architecture and are not intended to be limiting.

**Figure 13A** is a block diagram of a macro block that shows vertical edges 0-3. The shaded area represents data involved in a deblocking operation for edges 0 and 1, including data (on the far left) from a previous macro block. **Figure 13B** is a block diagram that shows the conceptual mapping of the shaded data from Figure 13A into the scratch buffer. In an embodiment, there are three scratch buffers that allow 16x3 pixels to fit in an area of 4x4 pixels, but other embodiments are possible within the scope of the claims. In an embodiment, there are three scratch buffer that allow 16x3 pixels to fit in an area of 4x4 pixels, but other embodiments are possible within the scope of the embodiments. In an embodiment deblocking mapping allows optimal use of four pipelines (Pipe 0, Pipe 1, Pipe 2, and Pipe 3) in the example architecture that has been previously described herein. However, the concepts described with reference to specific example architectures are equally applicable to other architectures not specifically described. For example, deblocking as described is also applicable or adaptable to future architectures (for example, 8x8 or 16x16) in which the screen tiling may not really exist.

**Figure 14A** is a block diagram that shows multiple macro blocks and their edges. Each of the macro blocks is similar to the single macro block shown in Figure 13A. Figure 14A shows the data involved in a single vertical deblocking pass according to an embodiment. **Figure 14B** is a block diagram that shows the mapping of the shaded data from Figure 14A into the scratch buffer in an arrangement that optimally uses the available hardware.

**Figure 15A** is a block diagram of a macro block that shows horizontal edges 0-3. The shaded area represents data involved in a deblocking operation for edge 0, including data (at the top) from a previous macro block. **Figure 15B** is a block diagram that shows the conceptual mapping of the shaded data from Figure 15A into the scratch buffer in an arrangement that optimally uses available pipelines in the example architecture that has been previously described herein.

**Figure 16A** is a block diagram that shows multiple macro blocks and their edges. Each macro block is similar to the single macro block shown in Figure 15A. The shaded data is the data involved in deblocking for edges 0. **Figure 16B** is a block diagram that shows the mapping of the shaded data from Figure 16A into the scratch buffer.
buffer in an arrangement that optimally uses the available hardware for performing deblocking on edges 0.

**Figure 17A** is a block diagram that shows multiple macro blocks and their edges. The shaded data is the data involved in deblocking for edges 0. **Figure 17B** is a block diagram that shows the mapping of the shaded data from Figure 17A into the scratch buffer in an arrangement that optimally uses the available hardware for performing deblocking on edges 1.

**Figure 18A** is a block diagram that shows multiple macro blocks and their edges. The shaded data is the data involved in deblocking for edges 2. **Figure 18B** is a block diagram that shows the mapping of the shaded data from Figure 18A into the scratch buffer in an arrangement that optimally uses the available hardware for performing deblocking on edges 2.

**Figure 19A** is a block diagram that shows multiple macro blocks and their edges. The shaded data is the data involved in deblocking for edges 3. **Figure 19B** is a block diagram that shows the mapping of the shaded data from Figure 19A into the scratch buffer in an arrangement that optimally uses the available hardware for performing deblocking on edges 3.

The mapping shown in Figures 13-19 is just one example of a mapping scheme for rearranging the pel data to be processed in a manner that optimizes the use of the available hardware.

Other variations on the methods and apparatus as described are also within the scope of the invention as claimed. For example, a scratch buffer could also be used in the inter-prediction and/or intra-prediction operations. Depending upon various factors, including the architecture of the graphics processing unit, using a scratch buffer may or may not be more efficient than processing "in place". In the embodiments described, which refer a particular architecture for the purpose of providing a coherent explanation, the deblocking operation benefits from using the scratch buffer. One reason is that the size and configuration of the pel data to be processed and the number of processing passes required do not vary. In addition, the order of the copies can vary. For example, copying can be done after every diagonal or after all of the diagonals. Therefore, the rearrangement for a particular architecture does not vary, and any performance penalties related to copying to the scratch buffer and copying back to the frame can be calculated. These performance penalties can be compared to the performance penalties associated with processing the pel data in place, but in
configurations that are not optimized for the hardware. An informed choice can then be made regarding whether to use the scratch buffer or not. On the other hand, for intra-prediction for example, the units of data to be processed are randomized by the encoding process, so it is not possible to accurately predict gains or losses associated with using the scratch buffer, and the overall performance over time may be about the same as for processing in place.

In another embodiment, the deblocking filtering is performed by a vertex shader for an entire macro block. In this regard the vertex shader works as a dedicated hardware pipeline. In various embodiments with different numbers of available pipelines, there may be four, eight or more available pipelines. In an embodiment, the deblocking algorithm involves two passes. The first pass is a vertical pass for all macro blocks along the diagonal being filtered (or deblocked). The second pass is a horizontal pass along the same diagonal.

The vertex shader process 256 pels of the luma macro block and 64 pels of each chroma macro block. In an embodiment, the vertex shader passes resulting filtered pels to pixel shaders through 16 parameter registers. Each register (128 bits) keeps one 4x4 filtered block of data. The "virtual pixel", or the pixel visible to the scan converter is an 8x8 block of pels for most of the passes. In an embodiment, eight render targets are defined. Each render target has a pixel format with two channels, and 32 bits per channel.

The pixel shader is invoked per 8x8 block. The pixel shader selects four proper registers from the 16 provided, rearranges them into eight 2x32-bit output color registers, and sends the data to the color buffer. In an embodiment, two buffers are used, a source buffer, and a target buffer. For this discussion, the target buffer is the scratch buffer. The source buffer is used as texture and the target is comprised of either four or eight render targets. The following tables illustrate buffer states during deblocking.

**Figures 20 and 21** show the state of the source buffer (Figure 20) and the target buffer (Figure 21) at the beginning of an algorithm iteration designated by the letter C. "C" marks the diagonal of the macro blocks to be filtered at the iteration C. "P" marks the previous diagonal. Both source buffer and target buffer keep the same data. Darkly shaded cells indicate already filtered macro blocks, white cells indicate not-yet-filtered macro blocks. Lightly shaded cells are partially filtered in the previous iteration. The iteration C consists of several passes.
Pass 1: Filtering the left side of the 0th vertical edge of each C macro block.
This pass is running along the P diagonal. Since the cell with an "X" in Figure 21 has no right neighbor, it is not a left neighbor itself and thus it is not taking part in this pass. A peculiarity of this pass is that the pixel shade: is invoked per 4x4 block and not per 8x8 block as in "standard" mode. 16 parameter registers are still sent to the pixel shader, but they are unpacked 32 bit float values. The target in this case has an ARGB type pixel format. There are 4 render targets. Figure 22 shows the state of the target buffer after the left side filtering.

Pass 2: Filtering vertical edges of each C macro block.
This pass is running along the C diagonal. During this pass the vertex/pixel shader pair is in a standard mode of operation. That is, the vertex shader sends 16 registers keeping a packed block of 4x4 pels each, and the pixel shader is invoked per 8x8 block, target pixel format (2 channel, 32 bit per channel). There are 8 render targets. Figure 23 shows the state of the target after the vertical filtering. After pass 2 the source and target are switched.

Pass 3: Copying the state of the P diagonal only from the new source (old target) to the new target (old source).
Figure 23 is a new source now. Figure 24 presents the state of the new target after the copy. In this pass the vertex shader does nothing. The pixel shader copies texture pixels in standard mode (format: 2 channels, 32 per channel, virtual pixel is 8x8) directly into the frame buffer. 8 render targets are involved.

Pass 4: Filtering the up side of the 0th horizontal edge of each C macro block.
This pass is running along the P diagonal. Since the cell with an "X" in Figure 24 has no down neighbor it is not an up neighbor itself and thus it is not taking part in the pass. Figure 25 represents the target state after the pass. It shows that the P diagonal is fully filtered inside the target frame buffer. The vertex/pixel shader pair works in the same mode as in pass 1.

Pass 5: Filtering horizontal edges of each C macro block.
This pass is running along the C diagonal. The resulting target is shown in Figure 26. Notice that, since the horizontal filter has been applied to the vertically filtered pels from the source (Figure 23), the target C cells are now both vertically and horizontally filtered.

After pass 2 the source and target are switched.
Pass 6: Copying the state of the P and C diagonals from the new source (old target) to the new target (old source).

Figure 26 is a new source. Figure 23 is a new target. Figure 27 is the state of the target after copy. The copying is done the same way as described with reference to Pass 3.

After making \( P = C \), and \( C = C + 1 \), the algorithm is ready for the next iteration.

Embodiments of the decoding method and system include a video data decoding method comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information; and decoding the encoded video data, wherein decoding comprises parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

In an embodiment the control information comprises control information specific to an architecture of a graphics processing unit (GPU).

In an embodiment the plurality of processing pipelines comprise a plurality of graphics processing unit (GPU) pipelines.

In an embodiment the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of data in a frame, and wherein the control information comprises an indication of which prediction operation was used to encode each unit of data in the frame.

An embodiment further comprises decoding the encoded video data on a frame basis such that each one of several decoding operations is performed on an entire frame of video data at a time.

In an embodiment the control information comprises a rearrangement of the video data such that a decoding operation can be performed in parallel on multiple video data using the plurality of GPU pipelines.

In an embodiment pre-processing further comprises creating a buffer from the control maps using one of a plurality of pre-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps, and wherein the buffer contains a subset of the control information.

In an embodiment the buffer is a Z-buffer.
In an embodiment the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H.264.

In an embodiment the pre-defined format comprises an MPEG standard video format.

An embodiment further comprises decoding the encoded video data on a frame basis such that each one of several decoding operations is performed on an entire frame of video data at a time.

In an embodiment the several decoding operations comprise inter-prediction, intra-prediction, and deblocking.

Embodiments of the decoding method and system further include a system for decoding video data encoded using a high-compression-ratio codec, the system comprising: a processing unit, comprising, a plurality of processing pipelines; and a driver comprising a layered decoder, wherein the layered decoder pre-processes control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, including control information specific to the plurality of processing pipelines.

An embodiment further comprises a Z-buffer coupled to the driver, wherein the Z-buffer is created from the control maps, and wherein generating the intermediate control maps comprises performing Z-testing on the Z-buffer.

In an embodiment the control information comprises information regarding rearranging the video data and directing the processing of the video data to be performed in parallel on the plurality of processing pipelines.

An embodiment further comprises a scratch buffer coupled to the driver, wherein the scratch buffer stores rearranged data for processing.

Embodiments of the decoding method and system further include a method for decoding video data encoded using a high-compression-ratio codec, the method comprising: pre-processing control maps that were generated during encoding of the video data; and generating intermediate control maps comprising information regarding decoding the video data on a frame basis such that each of multiple, distinct decoding operations is performed on an entire frame at one time, and further regarding rearranging the video data to be processed in parallel on multiple pipelines of a graphics processing unit (GPU) so as to optimize the use of the multiple pipelines.
An embodiment further comprises executing a plurality of setup passes on the control maps, comprising performing Z-testing of a Z-buffer created from the control maps.

An embodiment further comprises: determining from the intermediate control maps which data units within a frame are inter-prediction video data units; and performing inter-prediction on all of the inter-prediction data units within the frame.

An embodiment further comprises: determining from the intermediate control maps which data units within a frame are intra-prediction video data units; and performing intra-prediction on all of the intra-prediction video data units within the frame.

An embodiment further comprises: determining from the intermediate control maps video data units that do not have inter-unit dependencies for deblocking filtering, and rearranging the video data units that do not have inter-unit dependencies such that the data units that do not have inter-unit dependencies can be processed in parallel on the multiple pipelines.

An embodiment further comprises mapping the rearranged data units that do not have inter-unit dependencies to a scratch buffer for processing.

Embodiments of the decoding method and system further include a computer readable medium including instructions which when executed in a video processing system cause the system to process the encoded video data, the processing comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, including control information specific to an architecture of a video processing unit; and decoding the encoded video data, wherein decoding comprises parallel processing using the intermediate control maps to optimize usage of a plurality of graphics processing unit (GPU) pipelines.

In an embodiment the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of data in a frame, and wherein the control information comprises an indication of which prediction operation was used to encode each unit of data in the frame.
In an embodiment the processing further comprises decoding the encoded video data on a frame basis such that each one of several decoding operations is performed on an entire frame of video data at a time.

In an embodiment the control information comprises a rearrangement of the video data such that a decoding operation can be performed in parallel on multiple video data using the plurality of GPU pipelines.

In an embodiment pie-processing further comprises creating a Z-buffer from the control maps using one of a plurality of pre-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps.

In an embodiment the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H.264.

In an embodiment the pre-defined format comprises an MPEG standard video format.

In an embodiment the processing further comprises decoding the encoded video data on a frame basis such that each one of several decoding operations is performed on an entire frame of video data at a time.

In an embodiment the several decoding operations comprise inter-prediction, intra-prediction, and deblocking.

Embodiments of the decoding method and system further include a computer readable medium having instructions stored thereon which, when processed, are adapted to create a circuit capable of performing a method comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, including control information specific to an architecture of a video processing unit, and decoding the encoded video data, wherein decoding comprises parallel processing using the intermediate control maps to optimize usage of a plurality of graphics processing unit (GPU) pipelines.

Embodiments of the decoding method and system further include a computer having instructions store thereon which, when implemented in a video processing driver, cause the driver to perform a parallel processing method, the method comprising: pre-processing control maps that were generated from encoded video data; and generating intermediate control maps comprising information regarding decoding the video data on a frame basis such that each of multiple, distinct decoding operations is performed on an entire frame at one time, and further regarding rearranging the video
data to be processed in parallel on multiple pipelines of a graphics processing unit (GPU) so as to optimize the use of the multiple pipelines.

Embodiments of the decoding method and system further include a graphics processing unit (GPU) configured to: pre-process control maps that were generated from encoded video data; generate intermediate control maps; and use the intermediate control maps to perform decoding of the video data on a frame basis such that each of multiple, distinct decoding operations is performed on an entire frame at one time, and to further rearrange the video data to be processed in parallel on multiple pipelines of the GPU so as to optimize the use of the multiple pipelines.

Embodiments of the decoding method and system further include a video processing apparatus comprising: circuitry configured to pre-process control maps that were generated from encoded video data that was encoded according to a predefined format, and to generate intermediate control maps; and driver circuitry configured to read the intermediate control maps for controlling a video data decoding operation; and multiple video processing pipeline circuitry configured to respond to the driver circuitry to perform decoding of the video data on a frame basis such that each of multiple, distinct decoding operations is performed on an entire frame at one time, and to further rearrange the video data to be processed in parallel on multiple pipelines of the GPU so as to optimize the use of the multiple pipelines.

Embodiments include a digital image generated by a method comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information; and decoding the encoded video data, wherein decoding comprises parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

Embodiments of the decoding method and system further include a method for decoding video data, comprising: a first processor generating control maps from encoded video data; a second processor, receiving the control maps; generating intermediate control maps from the control maps, wherein the intermediate control maps include information specific to an architecture of the second processor; and using the intermediate control maps to decode the encoded video data.

In an embodiment the control maps comprise data and control information according to a specified format.
An embodiment further comprises the second processor using the intermediate control maps to perform parallel processing on the video data to generate display data.

In an embodiment control maps are generated on a per frame basis.

In an embodiment the architecture of the second processor comprises a type of architecture selected from a group comprising: a single instruction multiple data (SIMD) architecture; a multi-core architecture; and a multi-pipeline architecture.

In an embodiment parallel processing comprises performing set up passes.

In an embodiment performing setup passes comprises at least one of: sorting passes to sort surfaces; inter-prediction passes; intra-prediction passes; and deblocking passes.

Embodiments of the decoding method and system further include a method of upgrading a system to allow for decoding of video data comprising: causing an updated driver to be installed on the system, the updated driver containing computer readable instructions for adapting a system to pre-process control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information.

In an embodiment the computer readable instructions further adapt the system to decode the encoded video data, wherein decoding comprises parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

Embodiments of the decoding method and system further include a hardware-accelerated decoding method, comprising: pre-processing encoded data, wherein the encoded data is encoded in a plurality of units of predefined sizes, wherein various units of the plurality of units have dependencies such that dependent units must be processed in a particular order, and wherein pre-processing comprises determining the dependencies.

In an embodiment pre-processing further comprises designating units of data that have similar dependencies similarly, and processing similarly designated units in parallel.

In an embodiment designating units of data comprises: designating units of data that have similar inter-block dependencies similarly; and designating units of data that have similar intra-block dependencies similarly.
An embodiment further comprises: performing inter-prediction processing on similarly designated units of data in parallel; and performing intra-prediction processing on similarly designated units of data in parallel.

In an embodiment the method further comprises: decoding the preprocessed video data; performing further preprocessing on the decoded video data to determine deblocking dependencies; and designating units of decoded data having similar dependencies similarly.

An embodiment further comprises performing deblocking on multiple, similarly designated units in parallel.

Embodiments of the decoding method and system further include a video data decoding method comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, the plurality of prediction operations comprising inter-prediction; determining from the intermediate control maps which indicated units of video data are to be decoded using inter-prediction; performing inter-prediction on all of the indicated units of video data in the frame in parallel.

An embodiment further comprises performing inter-prediction on all of the indicated video data in multiple interleaved frames in parallel.

An embodiment further comprises parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

In an embodiment the plurality of processing pipelines comprise a plurality of graphics processing unit (GPU) pipelines.

In an embodiment the control information comprises a rearrangement of the video data such that a decoding operation can be performed in parallel on multiple video data using the plurality of GPU pipelines.

In an embodiment pre-processing further comprises creating a buffer from the control maps using one of a plurality of pre-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps, and wherein the buffer contains a subset of the control information.

In an embodiment the buffer is a Z-buffer.
In an embodiment determining comprises Z-testing to determine which of the plurality of prediction operations to perform on a unit of video data.

In an embodiment the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H.264.

In an embodiment the pre-defined format comprises an MPEG standard video format.

In an embodiment performing inter-prediction on all of the indicated units of video data comprises broadcasting information from the intermediate control maps to each indicated unit of video data.

An embodiment further comprises finding a reference frame using the information.

An embodiment further comprises finding reference pels within the reference frame using the information.

An embodiment further comprises combining reference pel data and residual data.

An embodiment further comprises writing a result for each indicated unit of data to a partially decoded frame.

Embodiments of the decoding method and system further include a system for decoding video data, the system comprising: a processing unit, comprising, a plurality of processing pipelines; and a driver comprising a layered decoder, wherein the layered decoder pre-processes control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information including, information indicating which of a plurality of prediction operations is to be performed on each unit of video data in a frame, the plurality of prediction operations comprising inter-prediction information regarding reference frames and reference pels; and control information specific to the plurality of processing pipelines.

An embodiment further comprises a Z-buffer coupled to the driver, wherein the Z-buffer is created from the control maps, and wherein generating the intermediate control maps comprises performing Z-testing on the Z-buffer to determine which units of video data in a frame are to be decoded using inter-prediction.

In an embodiment the control information comprises a rearrangement of the video data such that a decoding operation can be performed in parallel on multiple video data using the plurality of GPU pipelines.
Embodiments of the decoding method and system further include a method for decoding video data encoded using a high-compression-ratio codec, the method comprising: pre-processing control maps that were generated during encoding of the video data; and generating intermediate control maps comprising information regarding performing inter-prediction on the video data on a frame basis such inter-prediction is performed on an entire frame at one time.

An embodiment further comprises executing a plurality of setup passes on the control maps, comprising performing Z-testing of a Z-buffer created from the control maps, wherein at least one Z-buffer test indicates which of the units of video data to perform inter-prediction on.

Embodiments of the decoding method and system further include a computer readable medium including instructions which when executed in a video processing system cause the system to decode video data, the decoding comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, the plurality of prediction operations comprising inter-prediction; determining from the intermediate control maps which indicated units of video data are to decoded using inter-prediction; performing inter-prediction on all of the indicated units of video data in the frame in parallel.

In an embodiment the decoding further comprises performing inter-prediction on all of the indicated video data in multiple interleaved frames in parallel.

In an embodiment the decoding further comprises parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

In an embodiment the plurality of processing pipelines comprise a plurality of graphics processing unit (GPU) pipelines.

In an embodiment the control information comprises a rearrangement of the video data such that a decoding operation can be performed in parallel on multiple video data using the plurality of GPU pipelines.

In an embodiment pre-processing further comprises creating a Z-buffer from the control maps using one of a plurality of pre-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps.
In an embodiment determining comprises Z-testing to determine which of the plurality of prediction operations to perform on a unit of video data.

In an embodiment the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H 264.

In an embodiment the pre-defined format comprises an MPEG standard video format.

In an embodiment performing inter-prediction on all of the indicated units of video data comprises broadcasting information from the intermediate control maps to each indicated unit of video data.

In an embodiment performing inter-prediction further comprises finding a reference frame using the information.

In an embodiment performing inter-prediction further comprises finding reference pels within the reference frame using the information.

In an embodiment performing inter-prediction further comprises combining reference pel data and residual data.

In an embodiment performing inter-prediction further comprises writing a result for each indicated unit of data to a partially decoded frame.

Embodiments of the decoding method and system further include a computer readable medium having instructions stored thereon which, when processed, are adapted to create a circuit capable of performing a video data decoding method comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, the plurality of prediction operations comprising inter-prediction; determining from the intermediate control maps which indicated units of video data are to decoded using inter-prediction; performing inter-prediction on all of the indicated units of video data in the frame in parallel.

Embodiments of the decoding method and system further include a computer having instructions store thereon which, when implemented in a video processing driver, cause the driver to perform a parallel processing method, the method comprising: pre-processing control maps that were generated from encoded video data; and generating intermediate control maps comprising information regarding decoding.
the video data on a frame basis such that an inter-prediction operation is performed on an entire frame at one time.

Embodiments of the decoding method and system further include a graphics processing unit (GPU) configured to perform motion compensation, comprising: pre-processing control maps that were generated from encoded video data; generating intermediate control maps that indicate which units of video data in a frame are to be processed using an inter-prediction operation; and using the intermediate control maps to perform inter-prediction on the video data on a frame basis such that each inter-prediction is performed on an entire frame at one time, and to further rearrange the video data to be processed in parallel on multiple pipelines of the GPU so as to optimize the use of the multiple pipelines.

Embodiments of the decoding method and system further include a video processing apparatus comprising: circuitry configured to pre-process control maps that were generated from encoded video data that was encoded according to a predefined format, and to generate intermediate control maps that indicate which units of video data in a frame are to be processed using an inter-prediction operation; and driver circuitry configured to read the intermediate control maps for controlling a video data decoding operation, including performing the inter-prediction operation; and multiple video processing pipeline circuitry configured to respond to the driver circuitry to perform decoding of the video data on a frame basis such that the inter-prediction is performed on an entire frame at one time, and to further rearrange the video data to be processed in parallel on multiple pipelines of the GPU so as to optimize the use of the multiple pipelines.

Embodiments include a digital image generated by a method comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, the plurality of prediction operations comprising inter-prediction; determining from the intermediate control maps which indicated units of video data are to decoded using inter-prediction; performing inter-prediction on all of the indicated units of video data in the frame in parallel.
Embodiments of the decoding method and system further include a method for decoding video data, comprising, a first processor generating control maps from encoded video data; a second processor, receiving the control maps; generating intermediate control maps from the control maps, wherein the intermediate control maps indicate which units of video data in a frame are to be processed using an inter-prediction operation; and using the intermediate control maps to decode the encoded video data, comprising performing inter-prediction on all of the indicated units in the frame in parallel.

In an embodiment the intermediate control maps further comprise information specific to an architecture of the second processor.

In an embodiment the control maps comprise data and control information according to a specified video encoding format.

An embodiment further comprises the second processor using the intermediate control maps to perform parallel processing on the video data to generate display data.

In an embodiment control maps are generated on a per frame basis.

In an embodiment the architecture of the second processor comprises a type of architecture selected from a group comprising: a single instruction multiple data (SIMD) architecture, a multi-core architecture; and a multi-pipeline architecture.

In an embodiment parallel processing comprises performing setup passes.

In an embodiment performing setup passes comprises at least one of: sorting passes to sort surfaces, inter-prediction passes; and intra-prediction passes.

Embodiments of the decoding method and system further include a method of upgrading a system to allow for decoding of video data comprising: causing an updated driver to be installed on the system, the updated driver containing computer readable instructions for adapting a system to pre-process control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information indicating which units of video data in a frame are to be processed using an inter-prediction operation.

In an embodiment the computer readable instructions further adapt the system to decode the encoded video data, wherein decoding comprises parallel processing using the intermediate control maps to perform inter-prediction on all of the indicated units of data in the frame in parallel.
Embodiments of the decoding method and system further include a video data decoding method comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, the plurality of prediction operations comprising intra-prediction; determining from the intermediate control maps which indicated units of video data are to decoded using intra-prediction; performing intra-prediction on all of the indicated units of video data in the frame in parallel.

An embodiment further comprises performing intra-prediction on all of the indicated video data in multiple interleaved frames in parallel.

An embodiment further comprises parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

In an embodiment the plurality of processing pipelines comprise a plurality of graphics processing unit (GPU) pipelines.

In an embodiment the control information comprises designations for units of video data such that a decoding operation can be performed in parallel on similarly designated units of data using the plurality of GPU pipelines without errors due to inter-unit dependencies.

In an embodiment pre-processing further comprises creating a buffer from the control maps using one of a plurality of pre-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps, and wherein the buffer contains a subset of the control information.

In an embodiment the buffer is a Z-buffer.

In an embodiment determining comprises Z-testing to determine which of the plurality of prediction operations to perform on a unit of video data.

In an embodiment the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H.264.

In an embodiment the pre-defined format comprises an MPEG standard video format.

In an embodiment the control information comprises types of sub-units within the units of video data.
An embodiment further comprises similarly designating sub-units of video data to be processed concurrently using intra-prediction, wherein the similarly designated sub-units have similar inter-unit dependencies.

An embodiment further comprises arranging similarly designated sub-units of video data diagonally within the frame.

An embodiment further comprises running a shader on similarly designated sub-units of video data to perform intra-prediction on the similarly designated sub-units.

An embodiment further comprises writing a result for each indicated unit of data to a partially decoded frame.

Embodiments of the decoding method and system further include a system for decoding video data, the system comprising: a processing unit, comprising, a plurality of processing pipelines, and a driver comprising a layered decoder, wherein the layered decoder pre-processes control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information including, information indicating which of a plurality of prediction operations is to be performed on each unit of video data in a frame, the plurality of prediction operations comprising intra-prediction information regarding types of sub-units of video data within units of video data in the frame; and control information specific to the plurality of processing pipelines

An embodiment further comprises a Z-buffer coupled to the driver, wherein the Z-buffer is created from the control maps, and wherein generating the intermediate control maps comprises performing Z-testing on the Z-buffer to determine which units of video data in a frame are to be decoded using intra-prediction.

In an embodiment the control information comprises designations for units of video data such that a decoding operation can be performed in parallel on similarly designated units of data without errors due to inter-unit dependencies.

Embodiments of the decoding method and system further include a method for decoding video data encoded using a high-compression-ratio codec, the method comprising: pre-processing control maps that were generated during encoding of the video data; and generating intermediate control maps comprising information regarding performing intra-prediction on the video data on a frame basis such intra-prediction is performed on an entire frame at one time, and further regarding sub-units of video data
within the frame on which intra-prediction can be performed concurrently without errors due to dependencies between units of video data.

An embodiment further comprises executing a plurality of setup passes on the control maps, comprising performing Z-testing of a Z-buffer created from the control maps, wherein at least one Z-buffer test indicates which of the units of video data to perform intra-prediction on.

Embodiments of the decoding method and system further include a computer readable medium including instructions which when executed in a video processing system cause the system to decode video data, the decoding comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, the plurality of prediction operations comprising intra-prediction; determining from the intermediate control maps which indicated units of video data are to decoded using intra-prediction; performing intra-prediction on all of the indicated units of video data in the frame in parallel.

In an embodiment the decoding further comprises performing intra-prediction on all of the indicated video data in multiple interleaved frames in parallel.

In an embodiment the decoding further comprises parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

In an embodiment the plurality of processing pipelines comprise a plurality of graphics processing unit (GPU) pipelines.

In an embodiment the control information comprises designations for units of video data such that a decoding operation can be performed in parallel on similarly designated units of data using the plurality of GPU pipelines without errors due to inter-unit dependencies.

In an embodiment pre-processing further comprises creating a Z-buffer from the control maps using one of a plurality of pre-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps.

In an embodiment determining comprises Z-testing to determine which of the plurality of prediction operations to perform on a unit of video data.
In an embodiment the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H.264.

In an embodiment the pie-defined format comprises an MPEG standard video format.

In an embodiment the control information comprises types of sub-units within the units of video data.

In an embodiment decoding further comprises similarly designating sub-units of video data to be processed concurrently using intra-prediction, wherein the similarly designated sub-units have similar inter-unit dependencies.

In an embodiment decoding further comprises arranging similarly designated sub-units of video data diagonally within the frame.

In an embodiment decoding further comprises running a shader on similarly designated sub-units of video data to perform intra-prediction on the similarly designated sub-units.

In an embodiment performing intra-prediction further comprises writing a result for each indicated unit of data to a partially decoded frame.

Embodiments of the decoding method and system further include a computer readable medium having instructions stored thereon which, when processed, are adapted to create a circuit capable of performing a video data decoding method comprising, pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, the plurality of prediction operations comprising intra-prediction; determining from the intermediate control maps which indicated units of video data are to be decoded using intra-prediction; performing intra-prediction on all of the indicated units of video data in the frame in parallel.

Embodiments of the decoding method and system further include a computer having instructions store thereon which, when implemented in a video processing driver, cause the driver to perform a parallel processing method, the method comprising: pre-processing control maps that were generated from encoded video data, and generating intermediate control maps comprising information regarding decoding the video data on a frame basis such that an intra-prediction operation is performed on
an entire frame at one time, and further regarding groups of sub-units of video data in
the frame on which intra-prediction can be performed concurrently without errors due
to inter-unit dependencies.

Embodiments of the decoding method and system further include a graphics
processing unit (GPU) configured to perform motion compensation, comprising: pre-
processing control maps that were generated from encoded video data; generating
intermediate control maps that indicate which units of video data in a frame are to be
processed using an intra-prediction operation; and using the intermediate control maps
to perform intra-prediction on the video data on a frame basis such that each intra-
prediction is performed on an entire frame at one time, and to further rearrange the
video data to be processed in parallel on multiple pipelines of the GPU so as to
optimize the use of the multiple pipelines.

Embodiments of the decoding method and system further include a video
processing apparatus comprising: circuitry configured to pre-process control maps that
were generated from encoded video data that was encoded according to a predefined
format, and to generate intermediate control maps that indicate which units of video
data in a frame are to be processed using an intra-prediction operation; and driver
circuitry configured to read the intermediate control maps for controlling a video data
decoding operation, including performing the intra-prediction operation; and multiple
video processing pipeline circuitry configured to respond to the driver circuitry to
perform decoding of the video data on a frame basis such that the intra-prediction is
performed on an entire frame at one time, and to further rearrange the video data to be
processed in parallel on multiple pipelines of the GPU so as to optimize the use of the
multiple pipelines.

Embodiments include a digital image generated by a method comprising: pre-
processing control maps generated from encoded video data that was encoded
according to a pre-defined format, wherein pre-processing comprises generating a
plurality of intermediate control maps containing control information, and wherein the
pre-defined format comprises a compression scheme according to which the video data
may be encoded using one of a plurality of prediction operations for various units of
video data in a frame, the plurality of prediction operations comprising intra-prediction;
determining from the intermediate control maps which indicated units of video data are
to decoded using intra-prediction; performing intra-prediction on all of the indicated
units of video data in the same in parallel.
Embodiments of the decoding method and system further include a method for decoding video data, comprising: a first processor generating control maps from encoded video data; a second processor, receiving the control maps, generating intermediate control maps from the control maps, wherein the intermediate control maps indicate which units of video data in a frame are to be processed using an intra-prediction operation; and using the intermediate control maps to decode the encoded video data, comprising performing intra-prediction on all of the indicated units in the frame in parallel.

In an embodiment the intermediate control maps further comprise information specific to an architecture of the second processor.

In an embodiment the control maps comprise data and control information according to a specified video encoding format.

An embodiment further comprises the second processor using the intermediate control maps to perform parallel processing on the video data to generate display data.

In an embodiment control maps are generated on a per frame basis.

In an embodiment the architecture of the second processor comprises a type of architecture selected from a group comprising: a single instruction multiple data (SIMD) architecture; a multi-core architecture; and a multi-pipeline architecture.

In an embodiment parallel processing comprises performing setup passes.

In an embodiment performing setup passes comprises at least one of: sorting passes to sort surfaces; inter-prediction passes; and intra-prediction passes.

Embodiments of the decoding method and system further include a method of upgrading a system to allow for decoding of video data comprising: causing an updated driver to be installed on the system, the updated driver containing computer readable instructions for adapting a system to pre-process control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information indicating which units of video data in a frame are to be processed using an intra-prediction operation.

In an embodiment the computer readable instructions further adapt the system to decode the encoded video data, wherein decoding comprises parallel processing using the intermediate control maps to perform intra-prediction on all of the indicated units of data in the frame in parallel.
Embodiments of the decoding method and system further include a hardware-accelerated intra-prediction method, comprising: pre-processing encoded video data that is encoded in a plurality of units of predefined sizes, wherein various units of the plurality of units have dependencies such that dependent units must be processed in a particular order, and wherein pre-processing comprises determining the dependencies.

In an embodiment pre-processing further comprises designating units of data that have similar dependencies similarly, and performing intra-prediction on similarly designated units concurrently.

In an embodiment designating units of data comprises designating units of data that have similar inter-unit dependencies.

Embodiments of the decoding method and system further include a video data motion compensation method comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, and wherein the control information comprises an indication of which prediction operation was used to encode each unit of data in the frame; and decoding the encoded video data, wherein decoding comprises performing one of the indicated prediction operations in parallel on all of the video data in the frame encoded using the indicated prediction operations.

In an embodiment the method comprises performing one of the indicated prediction operations in parallel on all of the video data in multiple interleaved frames encoded using the indicated prediction operations.

In an embodiment the plurality of prediction operations comprise inter-prediction and intra-prediction.

In an embodiment decoding comprises parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

In an embodiment the plurality of processing pipelines comprise a plurality of graphics processing unit (GPU) pipelines.

In an embodiment the control information comprises a rearrangement of the video data such that a decoding operation can be performed in parallel on multiple video data using the plurality of GPU pipelines.
In an embodiment pre-processing further comprises creating a buffer from the control maps using one of a plurality of pre-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps, and wherein the buffer contains a subset of the control information.

In an embodiment the buffer is a Z-buffer.

In an embodiment decoding further comprises Z-testing to determine which of the plurality of prediction operations to perform on a unit of video data.

In an embodiment the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H.264.

In an embodiment the pre-defined format comprises an MPEG standard video format.

Embodiments of the decoding method and system further include a system for performing motion compensation in video data decoding, the system comprising: a processing unit, comprising, a plurality of processing pipelines; and a driver comprising a layered decoder, wherein the layered decoder pre-processes control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information including, information indicating which of a plurality of prediction operations is to be performed on each unit of video data in a frame; and control information specific to the plurality of processing pipelines.

In an embodiment, the system further comprises a Z-buffer coupled to the driver, wherein the Z-buffer is created from the control maps, and wherein generating the intermediate control maps comprises performing Z-testing on the Z-buffer to determine which of the plurality of prediction operations is to be performed on each unit of video data in a frame.

In an embodiment the control information comprises information regarding rearranging the video data and directing the processing of the video data to be performed in parallel on the plurality of processing pipelines, comprising performing one of the plurality of prediction operations on all of the units of video data indicated by the information.

Embodiments of the decoding method and system further include a method for motion compensation in decoding video data encoded using a high-compression-ratio codec, the method comprising: pre-processing control maps that were generated during encoding of the video data; and generating intermediate control maps comprising...
information regarding performing motion compensation on the video data on a frame basis such that each of multiple, distinct motion compensation operations is performed on an entire frame at one time, and further regarding rearranging the video data to be processed in parallel on multiple pipelines of a graphics processing unit (GPU) so as to optimize the use of the multiple pipelines.

In an embodiment, the multiple, distinct motion compensation operations comprise inter-prediction and intra-prediction.

An embodiment further comprises executing a plurality of setup passes on the control maps, comprising performing Z-testing of a Z-buffer created from the control maps, wherein at least one Z-buffer test indicates which of the multiple, distinct motion compensation operations is to be performed on each unit of video data in the frame.

An embodiment further comprises performing intra-prediction on all of the intra-prediction video data units within the frame.

An embodiment further comprises performing inter-prediction on all of the intra-prediction video data units within the frame.

Embodiments of the decoding method and system further include a computer readable medium including instructions which when executed in a video processing system cause the system to process encoded video data, including performing motion compensation, the processing comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, and wherein the control information comprises an indication of which prediction operation was used to encode each unit of data in the frame; and decoding the encoded video data, wherein decoding comprises performing one of the indicated prediction operations in parallel on all of the video data in the frame encoded using the indicated prediction operations.

In an embodiment, the processing further comprises, performing one of the indicated prediction operations in parallel on all of the video data in multiple interleaved frames encoded using the indicated prediction operations.

In an embodiment the plurality of prediction operations comprise inter-prediction and intra-prediction..
In an embodiment decoding comprises parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

In an embodiment the plurality of processing pipelines comprise a plurality of graphics processing unit (GPU) pipelines.

In an embodiment the control information comprises a rearrangement of the video data such that a decoding operation can be performed in parallel on multiple video data using the plurality of GPU pipelines.

In an embodiment pre-processing further comprises creating a Z-buffer from the control maps using one of a plurality of pre-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps.

In an embodiment decoding further comprises Z-testing to determine which of the plurality of prediction operations to perform on a unit of video data.

In an embodiment the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H.264.

In an embodiment the pre-defined format comprises an MPEG standard video format.

Embodiments of the decoding method and system further include a computer readable medium having instructions stored thereon which, when processed, are adapted to create a circuit capable of performing a motion compensation method comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, and wherein the control information comprises an indication of which prediction operation was used to encode each unit of data in the frame; and decoding the encoded video data, wherein decoding comprises performing one of the indicated prediction operations in parallel on all of the video data in the frame encoded using the indicated prediction operations.

Embodiments of the decoding method and system further include a computer having instructions store thereon which, when implemented in a video processing driver, cause the driver to perform a parallel processing method, the method comprising: pre-processing control maps that were generated from encoded video data; and generating intermediate control maps comprising information regarding decoding.
the video data on a frame basis such that each of multiple, distinct motion
compensation operations is performed on an entire frame at one time, and further
regarding rearranging the video data to be processed in parallel on multiple pipelines of
a graphics processing unit (GPU) so as to optimize the use of the multiple pipelines.

Embodiments of the decoding method and system further include a graphics
processing unit (GPU) configured to perform motion compensation, comprising: pre¬
processing control maps that were generated from encoded video data; generating
intermediate control maps that indicate which one of multiple prediction operations is
to be used in performing motion compensation on particular units of data in a frame;
and using the intermediate control maps to perform motion compensation on the video
data on a frame basis such that each of the multiple, distinct prediction operations is
performed on an entire frame at one time, and to further rearrange the video data to be
processed in parallel on multiple pipelines of the GPU so as to optimize the use of the
multiple pipelines.

Embodiments of the decoding method and system further include a video
processing apparatus comprising: circuitry configured to pre-process control maps that
were generated from encoded video data that was encoded according to a predefined
format, and to generate intermediate control maps that indicate which one of multiple
prediction operations is to be used in performing motion compensation on particular
units of data in a frame; and driver circuitry configured to read the intermediate control
maps for controlling a video data decoding operation, including performing one or
more of the multiple prediction operations; and multiple video processing pipeline
circuitry configured to respond to the driver circuitry to perform decoding of the video
data on a frame basis such that each of the multiple prediction operations is performed
on an entire frame at one time, and to further rearrange the video data to be processed
in parallel on multiple pipelines of the GPU so as to optimize the use of the multiple
pipelines.

A digital image generated by the method comprising: pre-processing control
maps generated from encoded video data that was encoded according to a pre-defined
format, wherein pre-processing comprises generating a plurality of intermediate control
maps containing control information, and wherein the pre-defined format comprises a
compression scheme according to which the video data may be encoded using one of a
plurality of prediction operations for various units of video data in a frame, and wherein
the control information comprises an indication of which prediction operation was used
to encode each unit of data in the frame; and decoding the encoded video data, wherein decoding comprises performing one of the indicated prediction operations in parallel on all of the video data in the frame encoded using the indicated prediction operations.

Embodiments of the decoding method and system further include a method for decoding video data, comprising: a first processor generating control maps from encoded video data, a second processor, receiving the control maps, generating intermediate control maps from the control maps, wherein the intermediate control maps indicate which one of multiple prediction operations is to be used in performing motion compensation on particular units of data in a frame; and using the intermediate control maps to decode the encoded video data, comprising performing motion compensation by performing an indicated prediction operation on all of the particular units in the frame in parallel.

In an embodiment the intermediate control maps further comprise information specific to an architecture of the second processor.

In an embodiment the control maps comprise data and control information according to a specified video encoding format.

An embodiment further comprises the second processor using the intermediate control maps to perform parallel processing on the video data to generate display data.

In an embodiment control maps are generated on a per frame basis.

In an embodiment the architecture of the second processor comprises a type of architecture selected from a group comprising: a single instruction multiple data (SIMD) architecture, a multi-core architecture; and a multi-pipeline architecture.

In an embodiment parallel processing comprises performing set up passes.

In an embodiment performing setup passes comprises at least one of: sorting passes to sort surfaces; inter-prediction passes; and intra-prediction passes.

Embodiments of the decoding method and system further include a method of upgrading a system to allow for decoding of video data comprising: causing an updated driver to be installed on the system, the updated driver containing computer readable instructions for adapting a system to pre-process control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information indicating which one of multiple prediction operations is to be used in performing motion compensation on particular units of data in a frame.
In an embodiment the computer readable instructions further adapt the system to decode the encoded video data, wherein decoding comprises parallel processing using the intermediate control maps to perform one of the indicated multiple prediction operations on all of the particular units of data in the frame in parallel.

Embodiments of the decoding method and system further include a hardware-accelerated motion compensation method, comprising: pre-processing encoded video data that is encoded in a plurality of units of predefined sizes, wherein various units of the plurality of units have dependencies such that dependent units must be processed in a particular order, and wherein pre-processing comprises determining the dependencies.

In an embodiment pre-processing further comprises:

- Designating units of data that have similar dependencies similarly, and
- Processing similarly designated units in parallel.

In an embodiment designating units of data comprises: designating units of data that have similar inter-unit dependencies similarly; and designating units of data that have similar intra-unit dependencies similarly.

An embodiment further comprises: performing inter-prediction processing on similarly designated units of data in parallel; and performing intra-prediction processing on similarly designated units of data in parallel.

Embodiments of the decoding method and system further include a video data decoding method comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information; and decoding the encoded video data, wherein decoding comprises: parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines; and performing deblocking on a frame of video data on which motion compensation has been performed.

In an embodiment the control information comprises control information specific to an architecture of a graphics processing unit (GPU).

In an embodiment the plurality of processing pipelines comprise a plurality of graphics processing unit (GPU) pipelines.

In an embodiment the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of data in a frame, and wherein the control information
comprises an indication of which prediction operation was used to encode each unit of data in the frame.

In an embodiment the control information comprises a rearrangement of the video data such that a decoding operation can be performed in parallel on multiple video data using the plurality of GPU pipelines

In an embodiment pre-processing further comprises creating a buffer from the control maps using one of a plurality of pre-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps, and wherein the buffer contains a subset of the control information.

In an embodiment the buffer is a Z-buffer.

In an embodiment the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H.264.

In an embodiment the pre-defined format comprises an MPEG standard video format.

An embodiment further comprises designating video data units in the frame on which one of vertical and horizontal deblocking can be performed concurrently.

An embodiment further comprises: mapping a plurality of similarly designated video data units to a scratch buffer such that the plurality of video data units is optimally processed by a particular architecture.

An embodiment further comprises: performing vertical deblocking on all of the similarly designated video data units, and performing horizontal deblocking on all of the similarly designated video data units.

Embodiments of the decoding method and system further include a system for decoding video data encoded using a high-compression-ratio codec, the system comprising: a processing unit, comprising, a plurality of processing pipelines; and a driver comprising a layered decoder, wherein the layered decoder pre-processes control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, including designations of video data macro blocks, wherein a similar designation indicates similar deblocking dependencies.

An embodiment further comprises a Z-buffer coupled to the driver, wherein the Z-buffer is created from the control maps, and wherein generating the intermediate control maps comprises performing Z-testing on the Z-buffer.
In an embodiment the control information comprises information regarding rearranging the video data and directing the processing of the video data to be performed in parallel on the plurality of processing pipelines.

An embodiment further comprises a scratch buffer coupled to the driver, wherein the scratch buffer stores rearranged data for processing.

Embodiments of the decoding method and system further include a method for decoding video data encoded using a high-compression-ratio codec, the method comprising: pre-processing control maps that were generated during encoding of the video data; and generating intermediate control maps comprising information regarding decoding the video data on a frame basis such that a deblocking operation is performed on an entire frame at one time, and further regarding rearranging the video data to be processed in parallel on multiple pipelines of a graphics processing unit (GPU) so as to optimize the use of the multiple pipelines.

An embodiment further comprises executing a plurality of setup passes on the control maps, comprising performing Z-testing of a Z-buffer created from the control maps.

An embodiment further comprises: determining from the intermediate control maps video data units that do not have inter-unit dependencies for deblocking filtering; and rearranging the video data units that do not have inter-unit dependencies such that the data units that do not have inter-unit dependencies can be processed in parallel on the multiple pipelines.

An embodiment further comprises mapping the rearranged data units that do not have inter-unit dependencies to a scratch buffer for processing.

Embodiments of the decoding method and system further include a computer readable medium including instructions which when executed in a video processing system cause the system to process the encoded video data, the processing comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information; and decoding the encoded video data, wherein decoding comprises: parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines; and performing deblocking on a frame of video data on motion compensation has been performed.
In an embodiment the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of data in a frame, and wherein the control information comprises an indication of which prediction operation was used to encode each unit of data in the frame.

In an embodiment the processing further comprises deblocking the decoded video data on a frame deblocking is performed on an entire frame of video data at a time.

In an embodiment the control information comprises a rearrangement of the video data such that a deblocking operation can be performed in parallel on multiple video data using the plurality of GPU pipelines.

In an embodiment the pre-processing further comprises creating a Z-buffer from the control maps using one of a plurality of pre-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps.

In an embodiment the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H.264.

In an embodiment the pre-defined format comprises an MPEG standard video format.

Embodiments of the decoding method and system further include a computer readable medium having instructions stored thereon which, when processed, are adapted to create a circuit capable of performing a method comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, including control information specific to an architecture of a video processing unit, and decoding the encoded video data; grouping units of video data that have similar deblocking dependencies; and performing deblocking on each group having the same dependencies concurrently.

Embodiments of the decoding method and system further include a computer having instructions store thereon which, when implemented in a video processing driver, cause the driver to perform a parallel processing method, the method comprising: pre-processing control maps that were generated from encoded video data; and generating intermediate control maps comprising information regarding decoding the video data on a frame basis such that each of multiple, distinct decoding operations,
including a deblocking operation, is performed on an entire frame at one time, and further regarding rearranging the video data to be processed in parallel on multiple pipelines of a graphics processing unit (GPU) so as to optimize the use of the multiple pipelines.

Embodiments of the decoding method and system further include a graphics processing unit (GPU) configured to: pre-process control maps that were generated from encoded video data; generate intermediate control maps; and use the intermediate control maps to perform deblocking of the video data on a frame basis such that deblocking is performed on an entire frame at one time, and to further rearrange the video data to be processed in parallel in groups of like dependencies on multiple pipelines of the GPU so as to optimize the use of the multiple pipelines.

Embodiments of the decoding method and system further include a video processing apparatus comprising: circuitry configured to pre-process control maps that were generated from encoded video data that was encoded according to a predefined format, and to generate intermediate control maps; and driver circuitry configured to read the intermediate control maps for controlling a video data decoding operation; and multiple video processing pipeline circuitry configured to respond to the driver circuitry to perform decoding of the video data on a frame basis such deblocking is performed on an entire frame at one time, and to further rearrange the video data to be processed in parallel in groups of like dependencies on multiple pipelines of the GPU so as to optimize the use of the multiple pipelines.

Embodiments of the decoding method and system further include a digital image generated by a method comprising: pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information; and decoding the encoded video data, wherein decoding comprises: parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines; and performing deblocking on a frame of video data on which motion compensation has been performed.

Embodiments of the decoding method and system further include a method for decoding video data, comprising: a first processor generating control maps from encoded video data; a second processor, receiving the control maps; generating intermediate control maps from the control maps, wherein the intermediate control maps include information specific to an architecture of the second processor; using the
intermediate control maps to decode the encoded video data; deblocking the decoded data, comprising deblocking an entire frame in parallel.

In an embodiment the control maps comprise data and control information according to a specified format

An embodiment further comprises the second processor using the intermediate control maps to perform parallel processing on the video data to generate display data.

In an embodiment control maps are generated on a per frame basis.

In an embodiment the architecture of the second processor comprises a type of architecture selected from a group comprising: a single instruction multiple data (SIMD) architecture; a multi-core architecture; and a multi-pipeline architecture.

In an embodiment parallel processing comprises performing setup passes.

In an embodiment performing setup passes comprises at least one of: sorting passes to sort surfaces; inter-prediction passes; intra-prediction passes; and deblocking passes.

Embodiments of the decoding method and system further include a method of upgrading a system to allow for decoding of video data comprising: causing an updated driver to be installed on the system, the updated driver containing computer readable instructions for adapting a system to pre-process control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information; and grouping units of data with similar deblocking dependencies such that a deblocking operation is performed on units in a group concurrently.

In an embodiment the computer readable instructions further adapt the system to decode the encoded video data, wherein decoding comprises parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

Embodiments of the decoding method and system further include a hardware-accelerated decoding method, comprising: pre-processing encoded data, wherein the encoded data is encoded in a plurality of units of predefined sizes, wherein various units of the plurality of units have dependencies, including deblocking dependencies, such that dependent units must be processed in a particular order, and wherein pre-processing comprises determining the dependencies; and performing deblocking on all of the units in a frame in one operation.
In an embodiment pre-processing further comprises designating units of data that have similar dependencies similarly, mapping units with similar dependencies to be processed together so as to optimally utilize the hardware; and processing similarly designated units in parallel.

In an embodiment the method further comprises: copying the mapped units to a buffer for processing; and copying the mapped units back to the frame after processing.

Aspects of the embodiments described above may be implemented as functionality programmed into any of a variety of circuitry, including but not limited to programmable logic devices (PLDs), such as field programmable gate arrays (FPGAs), programmable array logic (PAL) devices, electrically programmable logic and memory devices, and standard cell-based devices, as well as application specific integrated circuits (ASICs) and fully custom integrated circuits. Some other possibilities for implementing aspects of the embodiments include microcontrollers with memory (such as electronically erasable programmable read only memory (EEPROM)), embedded microprocessors, firmware, software, etc. Furthermore, aspects of the embodiments may be embodied in microprocessors having software-based circuit emulation, discrete logic (sequential and combinatorial), custom devices, fuzzy (neural) logic, quantum devices, and hybrids of any of the above device types. Of course the underlying device technologies may be provided in a variety of component types, e.g., metal-oxide semiconductor field-effect transistor (MOSFET) technologies such as complementary metal-oxide semiconductor (CMOS), bipolar technologies such as emitter-coupled logic (ECL), polymer technologies (e.g., silicon-conjugated polymer and metal-conjugated polymer-metal structures), mixed analog and digital, etc.

Unless the context clearly requires otherwise, throughout the description and the claims, the words "comprise," "comprising," and the like are to be construed in an inclusive sense as opposed to an exclusive or exhaustive sense, that is to say, in a sense of "including, but not limited to." Words using the singular or plural number also include the plural or singular number, respectively. Additionally, the words "herein," "hereunder," "above," "below," and words of similar import, when used in this application, refer to this application as a whole and not to any particular portions of this application. When the word "or" is used in reference to a list of two or more items, that word covers all of the following interpretations of the word, any of the items in the list, all of the items in the list, and any combination of the items in the list.
The above description of illustrated embodiments of the method and system is not intended to be exhaustive or to limit the invention to the precise forms disclosed. While specific embodiments of, and examples for, the method and system are described herein for illustrative purposes, various equivalent modifications are possible within the scope of the invention, as those skilled in the relevant art will recognize. The teachings of the disclosure provided herein can be applied to other systems, not only for systems including graphics processing or video processing, as described above. The various operations described may be performed in a very wide variety of architectures and distributed differently than described. In addition, though many configurations are described herein, none are intended to be limiting or exclusive.

In other embodiments, some or all of the hardware and software capability described herein may exist in a printer, a camera, television, a digital versatile disc (DVD) player, a handheld device, a mobile telephone or some other device. The elements and acts of the various embodiments described above can be combined to provide further embodiments. These and other changes can be made to the method and system in light of the above detailed description.

In general, in the following claims, the terms used should not be construed to limit the method and system to the specific embodiments disclosed in the specification and the claims, but should be construed to include any processing systems and methods that operate under the claims. Accordingly, the method and system is not limited by the disclosure, but instead the scope of the method and system is to be determined entirely by the claims.

While certain aspects of the method and system are presented below in certain claim forms, the inventors contemplate the various aspects of the method and system in any number of claim forms. For example, while only one aspect of the method and system may be recited as embodied in computer-readable medium, other aspects may likewise be embodied in computer-readable medium. Accordingly, the inventors reserve the right to add additional claims after filing the application to pursue such additional claim forms for other aspects of the method and system.
CLAIMS

What is claimed is:

1. A video data decoding method comprising:
   pre-processing control maps generated from encoded video data that was
   encoded according to a pie-defined format, wherein pre-processing comprises
   generating a plurality of intermediate control maps containing control information; and
   decoding the encoded video data, wherein decoding comprises parallel
   processing using the intermediate control maps to optimize usage of a plurality of
   processing pipelines.

2. The method of claim 1, wherein the control information comprises
   control information specific to an architecture of a graphics processing unit (GPU).

3. The method of claim 1, wherein the plurality of processing pipelines
   comprise a plurality of graphics processing unit (GPU) pipelines.

4. The method of claim 1, wherein the pre-defined format comprises a
   compression scheme according to which the video data may be encoded using one of a
   plurality of prediction operations for various units of data in a frame, and wherein the
   control information comprises an indication of which prediction operation was used to
   encode each unit of data in the frame.

5. The method of claim 1, further comprising decoding the encoded video
   data on a frame basis such that each one of several decoding operations is performed on
   an entire frame of video data at a time.

6. The method of claim 1, wherein the control information comprises a
   rearrangement of the video data such that a decoding operation can be performed in
   parallel on multiple video data using the plurality of GPU pipelines.

7. The method of claim 1, wherein pre-processing further comprises
   creating a buffer from the control maps using one of a plurality of pre-shaders, wherein
running a pie-shader on the control maps is more efficient than running a rendering shader on the control maps, and wherein the buffer contains a subset of the control information.

8. The method of claim 7, wherein the buffer is a Z-buffer.

9. The method of claim 4, wherein the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H 264

10. The method of claim 4, wherein the pre-defined format comprises an MPEG standard video format.

11. The method of claim 10, further comprising decoding the encoded video data on a frame basis such that each one of several decoding operations is performed on an entire frame of video data at a time.

12. The method of claim 11, wherein the several decoding operations comprise inter-prediction, intra-prediction, and deblocking.

13. A system for decoding video data encoded using a high-compression-ratio codec, the system comprising:
   a processing unit, comprising,
   a plurality of processing pipelines; and
   a driver comprising a layered decoder, wherein the layered decoder pre-processes control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, including control information specific to the plurality of processing pipelines.

14. The system of claim 13, further comprising a Z-buffer coupled to the driver, wherein the Z-buffer is created from the control maps, and wherein generating the intermediate control maps comprises performing Z-testing on the Z-buffer.
15. The system of claim 14, wherein the control information comprises information regarding rearranging the video data and directing the processing of the video data to be performed in parallel on the plurality of processing pipelines.

16. The system of claim 15, further comprising a scratch buffer coupled to the driver, wherein the scratch buffer stores rearranged data for processing.

17. A method for decoding video data encoded using a high-compression-ratio codec, the method comprising:
   pre-processing control maps that were generated during encoding of the video data; and
   generating intermediate control maps comprising information regarding decoding the video data on a frame basis such that each of multiple, distinct decoding operations is performed on an entire frame at one time, and further regarding rearranging the video data to be processed in parallel on multiple pipelines of a graphics processing unit (GPU) so as to optimize the use of the multiple pipelines.

18. The method of claim 17, further comprising executing a plurality of setup passes on the control maps, comprising performing Z-testing of a Z-buffer created from the control maps.

19. The method of 18, further comprising:
   determining from the intermediate control maps which data units within a frame are inter-prediction video data units; and
   performing inter-prediction on all of the inter-prediction data units within the frame.

20. The method of 18, further comprising:
   determining from the intermediate control maps which data units within a frame are intra-prediction video data units; and
   performing intra-prediction on all of the intra-prediction video data units within the frame.

21. The method of claim 20, further comprising:
determining from the intermediate control maps video data units that do not have inter-unit dependencies for deblocking filtering, and rearranging the video data units that do not have inter-unit dependencies such that the data units that do not have inter-unit dependencies can be processed in parallel on the multiple pipelines.

22. The method of claim 21, further comprising mapping the rearranged data units that do not have inter-unit dependencies to a scratch buffer for processing.

23. A computer readable medium including instructions which when executed in a video processing system cause the system to process the encoded video data, the processing comprising:

pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, including control information specific to an architecture of a video processing unit; and decoding the encoded video data, wherein decoding comprises parallel processing using the intermediate control maps to optimize usage of a plurality of graphics processing unit (GPU) pipelines.

24. The computer readable medium of claim 23, wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of data in a frame, and wherein the control information comprises an indication of which prediction operation was used to encode each unit of data in the frame.

25. The computer readable medium of claim 23, wherein the processing further comprises decoding the encoded video data on a frame basis such that each one of several decoding operations is performed on an entire frame of video data at a time.

26. The computer readable medium of claim 23, wherein the control information comprises a rearrangement of the video data such that a decoding operation can be performed in parallel on multiple video data using the plurality of GPU pipelines.
27. The computer readable medium of claim 23, wherein pre-processing further comprises creating a Z-buffer from the control maps using one of a plurality of pre-shaders, wherein running a pie-shader on the control maps is more efficient than running a rendering shader on the control maps.

28. The computer readable medium of claim 24, wherein the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H264.

29. The computer readable medium of claim 24, wherein the pre-defined format comprises an MPEG standard video format.

30. The computer readable medium of claim 23, wherein the processing further comprises decoding the encoded video data on a frame basis such that each one of several decoding operations is performed on an entire frame of video data at a time.

31. The computer readable medium of claim 30, wherein the several decoding operations comprise inter-prediction, intra-prediction, and deblocking.

32. A computer readable medium having instructions stored thereon which, when processed, are adapted to create a circuit capable of performing a method comprising:

   - pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, including control information specific to an architecture of a video processing unit; and
   - decoding the encoded video data, wherein decoding comprises parallel processing using the intermediate control maps to optimize usage of a plurality of graphics processing unit (GPU) pipelines.

33. A computer having instructions store thereon which, when implemented in a video processing driver, cause the driver to perform a parallel processing method, the method comprising:
pie-processing control maps that were generated from encoded video data; and

generating intermediate control maps comprising information regarding
decoding the video data on a frame basis such that each of multiple, distinct decoding
operations is performed on an entire frame at one time, and further regarding
rearranging the video data to be processed in parallel on multiple pipelines of a
graphics processing unit (GPU) so as to optimize the use of the multiple pipelines.

34 A graphics processing unit (GPU) configured to:
pre-process control maps that were generated from encoded video data;
generate intermediate control maps; and
use the intermediate control maps to perform decoding of the video data on a
frame basis such that each of multiple, distinct decoding operations is performed on an
entire frame at one time, and to further rearrange the video data to be processed in
parallel on multiple pipelines of the GPU so as to optimize the use of the multiple
pipelines.

35 A video processing apparatus comprising:
circuitry configured to pre-process control maps that were generated from
encoded video data that was encoded according to a predefined format, and to generate
intermediate control maps; and
driver circuitry configured to read the intermediate control maps for controlling
a video data decoding operation; and
multiple video processing pipeline circuitry configured to respond to the driver
circuitry to perform decoding of the video data on a frame basis such that each of
multiple, distinct decoding operations is performed on an entire frame at one time, and
to further rearrange the video data to be processed in parallel on multiple pipelines of
the GPU so as to optimize the use of the multiple pipelines.

36 A digital image generated by the method of claim 1.

37. A method for decoding video data, comprising:
a first processor generating control maps from encoded video data;
a second processor,
   receiving the control maps;
generating intermediate control maps from the control maps, wherein the intermediate control maps include information specific to an architecture of the second processor; and using the intermediate control maps to decode the encoded video data.

38. The method of claim 37, wherein the control maps comprise data and control information according to a specified format.

39. The method of claim 37, further comprising the second processor using the intermediate control maps to perform parallel processing on the video data to generate display data.

40. The method of claim 37, wherein control maps are generated on a per frame basis.

41. The method of claim 37, wherein the architecture of the second processor comprises a type of architecture selected from a group comprising:
   a single instruction multiple data (SIMD) architecture;
   a multi-core architecture; and
   a multi-pipeline architecture.

42. The method of claim 39, wherein parallel processing comprises performing setup passes.

43. The method of claim 42, wherein performing setup passes comprises at least one of:
   sorting passes to sort surfaces;
   inter-prediction passes;
   intra-prediction passes; and
   deblocking passes.

44. A method of upgrading a system to allow for decoding of video data comprising:
causing an updated driver to be installed on the system, the updated driver containing computer readable instructions for adapting a system to pre-process control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information.

45. The method of claim 44, wherein the computer readable instructions further adapt the system to decode the encoded video data, wherein decoding comprises parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

46. A hardware-accelerated decoding method, comprising:
pre-processing encoded data, wherein the encoded data is encoded in a plurality of units of predefined sizes, wherein various units of the plurality of units have dependencies such that dependent units must be processed in a particular order, and wherein pre-processing comprises determining the dependencies.

47. The method of claim 46, wherein pre-processing further comprises designating units of data that have similar dependencies similarly, and processing similarly designated units in parallel.

48. The method of claim 47, wherein designating units of data comprises:
designating units of data that have similar inter-block dependencies similarly; and
designating units of data that have similar intra-block dependencies similarly.

49. The method of claim 48, further comprising:
performing inter-prediction processing on similarly designated units of data in parallel; and
performing intra-prediction processing on similarly designated units of data in parallel.

50. The method of claim 46, wherein the method further comprises:
decoding the preprocessed video data;
performing further preprocessing on the decoded video data to determine
deblocking dependencies, and
designating units of decoded data having similar dependencies similarly.

51. The method of claim 50, further comprising performing deblocking on
multiple, similarly designated units in parallel.

52. A video data decoding method comprising:
pre-processing control maps generated from encoded video data that was
encoded according to a pre-defined format, wherein pre-processing comprises
generating a plurality of intermediate control maps containing control information, and
wherein the pre-defined format comprises a compression scheme according to which
the video data may be encoded using one of a plurality of prediction operations for
various units of video data in a frame, the plurality of prediction operations comprising
inter-prediction;
determining from the intermediate control maps which indicated units of video
data are to decoded using inter-prediction;
performing inter-prediction on all of the indicated units of video data in the
frame in parallel.

53. The method of claim 52, further comprising performing inter-prediction
on all of the indicated video data in multiple interleaved frames in parallel.

54. The method of claim 52, further comprising parallel processing using
the intermediate control maps to optimize usage of a plurality of processing pipelines.

55. The method of claim 54, wherein the plurality of processing pipelines
comprise a plurality of graphics processing unit (GPU) pipelines.

56. The method of claim 52, wherein the control information comprises a
rearrangement of the video data such that a decoding operation can be performed in
parallel on multiple video data using the plurality of GPU pipelines.
57. The method of claim 52, wherein pre-processing further comprises creating a buffer from the control maps using one of a plurality of pre-shaders, wherein running a pie-shader on the control maps is more efficient than running a rendering shader on the control maps, and wherein the buffer contains a subset of the control information.

58. The method of claim 57, wherein the buffer is a Z-buffer.

59. The method of claim 59, wherein determining comprises Z-testing to determine which of the plurality of prediction operations to perform on a unit of video data.

60. The method of claim 52, wherein the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H.264.

61. The method of claim 52, wherein the pre-defined format comprises an MPEG standard video format.

62. The method of claim 52, wherein performing inter-prediction on all of the indicated units of video data comprises broadcasting information from the intermediate control maps to each indicated unit of video data.

63. The method of claim 62, further comprising finding a reference frame using the information

64. The method of claim 63, further comprising finding reference pels within the reference frame using the information.

65. The method of claim 64, further comprising combining reference pel data and residual data.

66. The method of claim 65, further comprising writing a result for each indicated unit of data to a partially decoded frame.
67. A system for decoding video data, the system comprising:
   a processing unit, comprising,
   a plurality of processing pipelines; and
   a driver comprising a layered decoder, wherein the layered decoder pre-processes control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information including,
   information indicating which of a plurality of prediction operations is to be performed on each unit of video data in a frame, the plurality of prediction operations comprising inter-prediction
   information regarding reference frames and reference pels; and
   control information specific to the plurality of processing pipelines.

68. The system of claim 67, further comprising a Z-buffer coupled to the driver, wherein the Z-buffer is created from the control maps, and wherein generating the intermediate control maps comprises performing Z-testing on the Z-buffer to determine which units of video data in a frame are to be decoded using inter-prediction.

69. The system of claim 68, wherein the control information comprises a rearrangement of the video data such that a decoding operation can be performed in parallel on multiple video data using the plurality of GPU pipelines.

70. A method for decoding video data encoded using a high-compression-ratio codec, the method comprising:
   pre-processing control maps that were generated during encoding of the video data; and
   generating intermediate control maps comprising information regarding performing inter-prediction on the video data on a frame basis such inter-prediction is performed on an entire frame at one time.

71. The method of claim 70, further comprising executing a plurality of setup passes on the control maps, comprising performing Z-testing of a Z-buffer created from the control maps, wherein at least one Z-buffer test indicates which of the units of video data to perform inter-prediction on.
72. A computer readable medium including instructions which when executed in a video processing system cause the system to decode video data, the decoding comprising:

pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, the plurality of prediction operations comprising inter-prediction;

determining from the intermediate control maps which indicated units of video data are to be decoded using inter-prediction;

performing inter-prediction on all of the indicated units of video data in the frame in parallel.

73. The computer readable medium of claim 72, wherein the decoding further comprises performing inter-prediction on all of the indicated video data in multiple interleaved frames in parallel.

74. The computer readable medium of claim 72, wherein the decoding further comprises parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

75. The computer readable medium of claim 74, wherein the plurality of processing pipelines comprise a plurality of graphics processing unit (GPU) pipelines.

76. The computer readable medium of claim 72, wherein the control information comprises a rearrangement of the video data such that a decoding operation can be performed in parallel on multiple video data using the plurality of GPU pipelines.

77. The computer readable medium of claim 72, wherein pre-processing further comprises creating a Z-buffer from the control maps using one of a plurality of
pie-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps.

78. The computer readable medium of claim 77, wherein determining comprises Z-testing to determine which of the plurality of prediction operations to perform on a unit of video data

79. The computer readable medium of claim 72, wherein the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H.264.

80. The computer readable medium of claim 72, wherein the pre-defined format comprises an MPEG standard video format.

81. The computer readable medium of claim 72, wherein performing inter-prediction on all of the indicated units of video data comprises broadcasting information from the intermediate control maps to each indicated unit of video data.

82. The computer readable medium of claim 81, wherein performing inter-prediction further comprises finding a reference frame using the information

83. The computer readable medium of claim 82, wherein performing inter-prediction further comprises finding reference pels within the reference frame using the information.

84. The computer readable medium of claim 83, wherein performing inter-prediction further comprises combining reference pel data and residual data

85. The computer readable medium of claim 84, wherein performing inter-prediction further comprises writing a result for each indicated unit of data to a partially decoded frame.
86. A computer readable medium having instructions stored thereon which, when processed, are adapted to create a circuit capable of performing a video data decoding method comprising:

pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, the plurality of prediction operations comprising inter-prediction;

determining from the intermediate control maps which indicated units of video data are to be decoded using inter-prediction;

performing inter-prediction on all of the indicated units of video data in the frame in parallel.

87. A computer having instructions stored thereon which, when implemented in a video processing driver, cause the driver to perform a parallel processing method, the method comprising:

pre-processing control maps that were generated from encoded video data; and generating intermediate control maps comprising information regarding decoding the video data on a frame basis such that an inter-prediction operation is performed on an entire frame at one time.

88. A graphics processing unit (GPU) configured to perform motion compensation, comprising:

pre-processing control maps that were generated from encoded video data;

generating intermediate control maps that indicate which units of video data in a frame are to be processed using an inter-prediction operation; and

using the intermediate control maps to perform inter-prediction on the video data on a frame basis such that each inter-prediction is performed on an entire frame at one time, and to further rearrange the video data to be processed in parallel on multiple pipelines of the GPU so as to optimize the use of the multiple pipelines.

89. A video processing apparatus comprising-
circuitry configured to pre-process control maps that were generated from encoded video data that was encoded according to a predefined format, and to generate intermediate control maps that indicate which units of video data in a frame are to be processed using an inter-prediction operation; and

driver circuitry configured to read the intermediate control maps for controlling a video data decoding operation, including performing the inter-prediction operation; and

multiple video processing pipeline circuitry configured to respond to the driver circuitry to perform decoding of the video data on a frame basis such that the inter-prediction is performed on an entire frame at one time, and to further rearrange the video data to be processed in parallel on multiple pipelines of the GPU so as to optimize the use of the multiple pipelines.

90. A digital image generated by the method of claim 52.

91. A method for decoding video data, comprising:

a first processor generating control maps from encoded video data;
a second processor,

receiving the control maps;

generating intermediate control maps from the control maps, wherein the intermediate control maps indicate which units of video data in a frame are to be processed using an inter-prediction operation; and

using the intermediate control maps to decode the encoded video data, comprising performing inter-prediction on all of the indicated units in the frame in parallel.

92. The method of claim 91, wherein the intermediate control maps further comprise information specific to an architecture of the second processor.

93. The method of claim 91, wherein the control maps comprise data and control information according to a specified video encoding format.
The method of claim 92, further comprising the second processor using the intermediate control maps to perform parallel processing on the video data to generate display data.

The method of claim 92, wherein control maps are generated on a per frame basis.

The method of claim 92, wherein the architecture of the second processor comprises a type of architecture selected from a group comprising:
- a single instruction multiple data (SIMD) architecture;
- a multi-core architecture; and
- a multi-pipeline architecture.

The method of claim 94, wherein parallel processing comprises performing setup passes.

The method of claim 97, wherein performing setup passes comprises at least one of:
- sorting passes to sort surfaces;
- inter-prediction passes; and
- intra-prediction passes.

A method of upgrading a system to allow for decoding of video data comprising:
causing an updated driver to be installed on the system, the updated driver containing computer readable instructions for adapting a system to pre-process control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information indicating which units of video data in a frame are to be processed using an inter-prediction operation.

The method of claim 99, wherein the computer readable instructions further adapt the system to decode the encoded video data, wherein decoding comprises parallel processing using the intermediate control maps to perform inter-prediction on all of the indicated units of data in the frame in parallel.
101. A video data decoding method comprising:

pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, the plurality of prediction operations comprising intra-prediction;

determining from the intermediate control maps which indicated units of video data are to be decoded using intra-prediction;

performing intra-prediction on all of the indicated units of video data in the frame in parallel.

102. The method of claim 101, further comprising performing intra-prediction on all of the indicated video data in multiple interleaved frames in parallel.

103. The method of claim 101, further comprising parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

104. The method of claim 103, wherein the plurality of processing pipelines comprise a plurality of graphics processing unit (GPU) pipelines.

105. The method of claim 101, wherein the control information comprises designations for units of video data such that a decoding operation can be performed in parallel on similarly designated units of data using the plurality of GPU pipelines without errors due to inter-unit dependencies.

106. The method of claim 101, wherein pre-processing further comprises creating a buffer from the control maps using one of a plurality of pre-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps, and wherein the buffer contains a subset of the control information.
107. The method of claim 106, wherein the buffer is a Z-buffer.

108. The method of claim 107, wherein determining comprises Z-testing to determine which of the plurality of prediction operations to perform on a unit of video data.

109. The method of claim 101, wherein the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H.264.

110. The method of claim 101, wherein the pre-defined format comprises an MPEG standard video format.

111. The method of claim 101, wherein the control information comprises types of sub-units within the units of video data.

112. The method of claim 111, further comprising similarly designating sub-units of video data to be processed concurrently using intra-prediction, wherein the similarly designated sub-units have similar inter-unit dependencies.

113. The method of claim 112, further comprising arranging similarly designated sub-units of video data diagonally within the frame.

114. The method of claim 113, further comprising running a shader on similarly designated sub-units of video data to perform intra-prediction on the similarly designated sub-units.

115. The method of claim 112, further comprising writing a result for each indicated unit of data to a partially decoded frame.

116. A system for decoding video data, the system comprising:
   a processing unit, comprising,
   a plurality of processing pipelines; and
   a driver comprising a layered decoder, wherein the layered decoder pre-processes control maps generated from encoded video data that was encoded according
to a pie-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information including,

- information indicating which of a plurality of prediction operations is to be performed on each unit of video data in a frame, the plurality of prediction operations comprising intra-prediction
- information regarding types of sub-units of video data within units of video data in the frame; and
- control information specific to the plurality of processing pipelines.

117. The system of claim 116, further comprising a Z-buffer coupled to the driver, wherein the Z-buffer is created from the control maps, and wherein generating the intermediate control maps comprises performing Z-testing on the Z-buffer to determine which units of video data in a frame are to be decoded using intra-prediction.

118. The system of claim 117, wherein the control information comprises designations for units of video data such that a decoding operation can be performed in parallel on similarly designated units of data without errors due to inter-unit dependencies.

119. A method for decoding video data encoded using a high-compression-ratio codec, the method comprising:

- pre-processing control maps that were generated during encoding of the video data, and

- generating intermediate control maps comprising information regarding performing intra-prediction on the video data on a frame basis such intra-prediction is performed on an entire frame at one time, and further regarding sub-units of video data within the frame on which intra-prediction can be performed concurrently without errors due to dependencies between units of video data.

120. The method of claim 119, further comprising executing a plurality of setup passes on the control maps, comprising performing Z-testing of a Z-buffer created from the control maps, wherein at least one Z-buffer test indicates which of the units of video data to perform intra-prediction on.
121. A computer readable medium including instructions which when executed in a video processing system cause the system to decode video data, the decoding comprising:

pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, the plurality of prediction operations comprising intra-prediction,

determining from the intermediate control maps which indicated units of video data are to decoded using intra-prediction;

performing intra-prediction on all of the indicated units of video data in the frame in parallel.

122. The computer readable medium of claim 121, wherein the decoding further comprises performing intra-prediction on all of the indicated video data in multiple interleaved frames in parallel.

123. The computer readable medium of claim 121, wherein the decoding further comprises parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

124. The computer readable medium of claim 123, wherein the plurality of processing pipelines comprise a plurality of graphics processing unit (GPU) pipelines.

125. The computer readable medium of claim 121, wherein the control information comprises designations for units of video data such that a decoding operation can be performed in parallel on similarly designated units of data using the plurality of GPU pipelines without errors due to inter-unit dependencies.

126. The computer readable medium of claim 121, wherein pre-processing further comprises creating a Z-buffer from the control maps using one of a plurality of
pie-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps.

127. The computer readable medium of claim 126, wherein determining comprises Z-testing to determine which of the plurality of prediction operations to perform on a unit of video data.

128. The computer readable medium of claim 121, wherein the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H 264.

129. The computer readable medium of claim 121, wherein the pre-defined format comprises an MPEG standard video format.

130. The computer readable medium of claim 121, wherein the control information comprises types of sub-units within the units of video data.

131. The computer readable medium of claim 130, wherein decoding further comprises similarly designating sub-units of video data to be processed concurrently using intra-prediction, wherein the similarly designated sub-units have similar inter-unit dependencies.

132. The computer readable medium of claim 131, wherein decoding further comprises arranging similarly designated sub-units of video data diagonally within the frame.

133. The computer readable medium of claim 132, wherein decoding further comprises running a shader on similarly designated sub-units of video data to perform intra-prediction on the similarly designated sub-units.

134. The computer readable medium of claim 133, wherein performing intra-prediction further comprises writing a result for each indicated unit of data to a partially decoded frame.
135 A computer readable medium having instructions stored thereon which, when processed, are adapted to create a circuit capable of performing a video data decoding method comprising:

pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, the plurality of prediction operations comprising intra-prediction,

determining from the intermediate control maps which indicated units of video data are to be decoded using intra-prediction;

performing intra-prediction on all of the indicated units of video data in the frame in parallel

136. A computer having instructions store thereon which, when implemented in a video processing driver, cause the driver to perform a parallel processing method, the method comprising:

pre-processing control maps that were generated from encoded video data; and

generating intermediate control maps comprising information regarding decoding the video data on a frame basis such that an intra-prediction operation is performed on an entire frame at one time, and further regarding groups of sub-units of video data in the frame on which intra-prediction can be performed concurrently without errors due to inter-unit dependencies.

137. A graphics processing unit (GPU) configured to perform motion compensation, comprising:

pre-processing control maps that were generated from encoded video data;

generating intermediate control maps that indicate which units of video data in a frame are to be processed using an intra-prediction operation; and

using the intermediate control maps to perform intra-prediction on the video data on a frame basis such that each intra-prediction is performed on an entire frame at one time, and to further rearrange the video data to be processed in parallel on multiple pipelines of the GPU so as to optimize the use of the multiple pipelines.
138. A video processing apparatus comprising:

circuitry configured to pre-process control maps that were generated from encoded video data that was encoded according to a predefined format, and to generate intermediate control maps that indicate which units of video data in a frame are to be processed using an intra-prediction operation; and

driver circuitry configured to read the intermediate control maps for controlling a video data decoding operation, including performing the intra-prediction operation; and

multiple video processing pipeline circuitry configured to respond to the driver circuitry to perform decoding of the video data on a frame basis such that the intra-prediction is performed on an entire frame at one time, and to further rearrange the video data to be processed in parallel on multiple pipelines of the GPU so as to optimize the use of the multiple pipelines.

139. A digital image generated by the method of claim 101.

140. A method for decoding video data, comprising:

a first processor generating control maps from encoded video data;
a second processor,

receiving the control maps;

generating intermediate control maps from the control maps, wherein the intermediate control maps indicate which units of video data in a frame are to be processed using an intra-prediction operation; and

using the intermediate control maps to decode the encoded video data, comprising performing intra-prediction on all of the indicated units in the frame in parallel.

141. The method of claim 140, wherein the intermediate control maps further comprise information specific to an architecture of the second processor.

142. The method of claim 140, wherein the control maps comprise data and control information according to a specified video encoding format.
143. The method of claim 141, farther comprising the second processor using the intermediate control maps to perform parallel processing on the video data to generate display data.

144. The method of claim 141, wherein control maps are generated on a per frame basis.

145. The method of claim 141, wherein the architecture of the second processor comprises a type of architecture selected from a group comprising:
   a single instruction multiple data (SIMD) architecture;
   a multi-core architecture; and
   a multi-pipeline architecture.

146. The method of claim 143, wherein parallel processing comprises performing setup passes.

147. The method of claim 146, wherein performing setup passes comprises at least one of:
    sorting passes to sort surfaces;
    inter-prediction passes; and
    intra-prediction passes.

148. A method of upgrading a system to allow for decoding of video data comprising:
    causing an updated driver to be installed on the system, the updated driver containing computer readable instructions for adapting a system to pre-process control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information indicating which units of video data in a frame are to be processed using an intra-prediction operation.

149. The method of claim 148, wherein the computer readable instructions further adapt the system to decode the encoded video data, wherein decoding comprises
parallel processing using the intermediate control maps to perform intra-prediction on all of the indicated units of data in the frame in parallel.

150. A hardware-accelerated intra-prediction method, comprising:

pre-processing encoded video data that is encoded in a plurality of units of predefined sizes, wherein various units of the plurality of units have dependencies such that dependent units must be processed in a particular order, and wherein pre-processing comprises determining the dependencies.

151. The method of claim 150, wherein pre-processing further comprises designating units of data that have similar dependencies similarly, and performing intra-prediction on similarly designated units concurrently.

152. The method of claim 150, wherein designating units of data comprises designating units of data that have similar inter-unit dependencies similarly.

153. A video data motion compensation method comprising:

pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, and wherein the control information comprises an indication of which prediction operation was used to encode each unit of data in the frame; and

decoding the encoded video data, wherein decoding comprises performing one of the indicated prediction operations in parallel on all of the video data in the frame encoded using the indicated prediction operations.

154. The method of claim 153, further comprising performing one of the indicated prediction operations in parallel on all of the video data in multiple interleaved frames encoded using the indicated prediction operations.
155. The method of claim 153, wherein the plurality of prediction operations comprise inter-prediction and intra-prediction.

156. The method of claim 153, wherein decoding comprises parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

157. The method of claim 156, wherein the plurality of processing pipelines comprise a plurality of graphics processing unit (GPU) pipelines.

158. The method of claim 153, wherein the control information comprises a rearrangement of the video data such that a decoding operation can be performed in parallel on multiple video data using the plurality of GPU pipelines.

159. The method of claim 153, wherein pre-processing further comprises creating a buffer from the control maps using one of a plurality of pre-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps, and wherein the buffer contains a subset of the control information.

160. The method of claim 159, wherein the buffer is a Z-buffer.

161. The method of claim 160, wherein decoding further comprises Z-testing to determine which of the plurality of prediction operations to perform on a unit of video data.

162. The method of claim 153, wherein the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H.264.

163. The method of claim 153, wherein the pre-defined format comprises an MPEG standard video format.

164. A system for performing motion compensation in video data decoding, the system comprising:
a processing unit, comprising,
a plurality of processing pipelines; and
a driver comprising a layered decoder, wherein the layered decoder pre-processes control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information including,
information indicating which of a plurality of prediction operations is to be performed on each unit of video data in a frame; and
control information specific to the plurality of processing pipelines.

165. The system of claim 164, further comprising a Z-buffer coupled to the driver, wherein the Z-buffer is created from the control maps, and wherein generating the intermediate control maps comprises performing Z-testing on the Z-buffer to determine which of the plurality of prediction operations is to be performed on each unit of video data in a frame.

166. The system of claim 165, wherein the control information comprises information regarding rearranging the video data and directing the processing of the video data to be performed in parallel on the plurality of processing pipelines, comprising performing one of the plurality of prediction operations on all of the units of video data indicated by the information.

167. A method for motion compensation in decoding video data encoded using a high-compression-ratio codec, the method comprising:
pre-processing control maps that were generated during encoding of the video data; and
generating intermediate control maps comprising information regarding performing motion compensation on the video data on a frame basis such that each of multiple, distinct motion compensation operations is performed on an entire frame at one time, and further regarding rearranging the video data to be processed in parallel on multiple pipelines of a graphics processing unit (GPU) so as to optimize the use of the multiple pipelines.
168. The method of claim 167, wherein the multiple, distinct motion compensation operations comprise inter-prediction and intra-prediction.

169. The method of claim 168, further comprising executing a plurality of setup passes on the control maps, comprising performing Z-testing of a Z-buffer created from the control maps, wherein at least one Z-buffer test indicates which of the multiple, distinct motion compensation operations is to be performed on each unit of video data in the frame.

170. The method of 169, further comprising performing intra-prediction on all of the intra-prediction video data units within the frame.

171. The method of 169, further comprising performing inter-prediction on all of the intra-prediction video data units within the frame.

172. A computer readable medium including instructions which when executed in a video processing system cause the system to process encoded video data, including performing motion compensation, the processing comprising:
    pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, and wherein the control information comprises an indication of which prediction operation was used to encode each unit of data in the frame, and
    decoding the encoded video data, wherein decoding comprises performing one of the indicated prediction operations in parallel on all of the video data in the frame encoded using the indicated prediction operations.

173. The computer readable medium of claim 172, wherein the processing further comprises, performing one of the indicated prediction operations in parallel on all of the video data in multiple interleaved frames encoded using the indicated prediction operations.
174. The computer readable medium of claim 172, wherein the plurality of prediction operations comprise inter-prediction and intra-prediction.

175. The computer readable medium of claim 172, wherein decoding comprises parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

176. The computer readable medium of claim 175, wherein the plurality of processing pipelines comprise a plurality of graphics processing unit (GPU) pipelines.

177. The computer readable medium of claim 172, wherein the control information comprises a rearrangement of the video data such that a decoding operation can be performed in parallel on multiple video data using the plurality of GPU pipelines.

178. The computer readable medium of claim 172, wherein pre-processing further comprises creating a Z-buffer from the control maps using one of a plurality of pre-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps.

179. The computer readable medium of claim 178, wherein decoding further comprises Z-testing to determine which of the plurality of prediction operations to perform on a unit of video data.

180. The computer readable medium of claim 174, wherein the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H.264.

181. The computer readable medium of claim 172, wherein the pre-defined format comprises an MPEG standard video format.
182. A computer readable medium having instructions stored thereon which, when processed, are adapted to create a circuit capable of performing a motion compensation method comprising:

pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, and wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of video data in a frame, and wherein the control information comprises an indication of which prediction operation was used to encode each unit of data in the frame; and

decoding the encoded video data, wherein decoding comprises performing one of the indicated prediction operations in parallel on all of the video data in the frame encoded using the indicated prediction operations

183. A computer having instructions stored thereon which, when implemented in a video processing driver, cause the driver to perform a parallel processing method, the method comprising:

pre-processing control maps that were generated from encoded video data; and

generating intermediate control maps comprising information regarding decoding the video data on a frame basis such that each of multiple, distinct motion compensation operations is performed on an entire frame at one time, and further regarding rearranging the video data to be processed in parallel on multiple pipelines of a graphics processing unit (GPU) so as to optimize the use of the multiple pipelines

184. A graphics processing unit (GPU) configured to perform motion compensation, comprising:

pre-processing control maps that were generated from encoded video data;

generating intermediate control maps that indicate which one of multiple prediction operations is to be used in performing motion compensation on particular units of data in a frame; and

using the intermediate control maps to perform motion compensation on the video data on a frame basis such that each of the multiple, distinct prediction operations is performed on an entire frame at one time, and to further rearrange the video data to
be processed in parallel on multiple pipelines of the GPU so as to optimize the use of the multiple pipelines.

185. A video processing apparatus comprising:
circuitry configured to pre-process control maps that were generated from encoded video data that was encoded according to a predefined format, and to generate intermediate control maps that indicate which one of multiple prediction operations is to be used in performing motion compensation on particular units of data in a frame; and
driver circuitry configured to read the intermediate control maps for controlling a video data decoding operation, including performing one or more of the multiple prediction operations; and
multiple video processing pipeline circuitry configured to respond to the driver circuitry to perform decoding of the video data on a frame basis such that each of the multiple prediction operations is performed on an entire frame at one time, and to further rearrange the video data to be processed in parallel on multiple pipelines of the GPU so as to optimize the use of the multiple pipelines.

186. A digital image generated by the method of claim 153.

187. A method for decoding video data, comprising:
a first processor generating control maps from encoded video data;
a second processor,
receiving the control maps;
generating intermediate control maps from the control maps, wherein the intermediate control maps indicate which one of multiple prediction operations is to be used in performing motion compensation on particular units of data in a frame; and
using the intermediate control maps to decode the encoded video data, comprising performing motion compensation by performing an indicated prediction operation on all of the particular units in the frame in parallel.

188

The method of claim 187, wherein the intermediate control maps further comprise information specific to an architecture of the second processor.
189. The method of claim 187, wherein the control maps comprise data and control information according to a specified video encoding format.

190. The method of claim 188, further comprising the second processor using the intermediate control maps to perform parallel processing on the video data to generate display data.

191. The method of claim 188, wherein control maps are generated on a per frame basis.

192. The method of claim 188, wherein the architecture of the second processor comprises a type of architecture selected from a group comprising:
   a single instruction multiple data (SIMD) architecture;
   a multi-core architecture; and
   a multi-pipeline architecture.

193. The method of claim 190, wherein parallel processing comprises performing setup passes.

194. The method of claim 193, wherein performing setup passes comprises at least one of:
   sorting passes to sort surfaces,
   inter-prediction passes; and
   intra-prediction passes.

195. A method of upgrading a system to allow for decoding of video data comprising:
   causing an updated driver to be installed on the system, the updated driver containing computer readable instructions for adapting a system to pre-process control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information indicating which one of multiple prediction
operations is to be used in performing motion compensation on particular units of data in a frame.

196. The method of claim 195, wherein the computer readable instructions further adapt the system to decode the encoded video data, wherein decoding comprises parallel processing using the intermediate control maps to perform one of the indicated multiple prediction operations on all of the particular units of data in the frame in parallel.

197. A hardware-accelerated motion compensation method, comprising:
pre-processing encoded video data that is encoded in a plurality of units of predefined sizes, wherein various units of the plurality of units have dependencies such that dependent units must be processed in a particular order, and wherein pre-processing comprises determining the dependencies.

198. The method of claim 197, wherein pre-processing further comprises:
designating units of data that have similar dependencies similarly, and processing similarly designated units in parallel.

199. The method of claim 198, wherein designating units of data comprises:
designating units of data that have similar inter-unit dependencies similarly; and designating units of data that have similar intra-unit dependencies similarly.

200. The method of claim 199, further comprising:
performing inter-prediction processing on similarly designated units of data in parallel; and
performing intra-prediction processing on similarly designated units of data in parallel.

201. A video data decoding method comprising:
pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information; and decoding the encoded video data, wherein decoding comprises:
parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines; and performing deblending on a frame of video data on which motion compensation has been performed.

202. The method of claim 201, wherein the control information comprises control information specific to an architecture of a graphics processing unit (GPU).

203. The method of claim 201, wherein the plurality of processing pipelines comprise a plurality of graphics processing unit (GPU) pipelines.

204. The method of claim 201, wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of data in a frame, and wherein the control information comprises an indication of which prediction operation was used to encode each unit of data in the frame.

205. The method of claim 201, wherein the control information comprises a rearrangement of the video data such that a decoding operation can be performed in parallel on multiple video data using the plurality of GPU pipelines.

206. The method of claim 201, wherein pre-processing further comprises creating a buffer from the control maps using one of a plurality of pre-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps, and wherein the buffer contains a subset of the control information.

207. The method of claim 206, wherein the buffer is a Z-buffer.

208. The method of claim 204, wherein the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H.264.

209. The method of claim 204, wherein the pre-defined format comprises an MPEG standard video format.
210. The method of claim 208, further comprising designating video data units in the frame on which one of vertical and horizontal deblocking can be performed concurrently.

211. The method of claim 210, further comprising:
    mapping a plurality of similarly designated video data units to a scratch buffer such that the plurality of video data units is optimally processed by a particular architecture.

212. The method of claim 211, further comprising:
    performing vertical deblocking on all of the similarly designated video data units; and
    performing horizontal deblocking on all of the similarly designated video data units.

213. A system for decoding video data encoded using a high-compression-ratio codec, the system comprising:
    a processing unit, comprising,
    a plurality of processing pipelines; and
    a driver comprising a layered decoder, wherein the layered decoder pre-processes control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, including designations of video data macro blocks, wherein a similar designation indicates similar deblocking dependencies.

214. The system of claim 213, further comprising a Z-buffer coupled to the driver, wherein the Z-buffer is created from the control maps, and wherein generating the intermediate control maps comprises performing Z-testing on the Z-buffer.

215. The system of claim 214, wherein the control information comprises information regarding rearranging the video data and directing the processing of the video data to be performed in parallel on the plurality of processing pipelines
216. The system of claim 215, further comprising a scratch buffer coupled to the driver, wherein the scratch buffer stores rearranged data for processing.

217. A method for decoding video data encoded using a high-compression-ratio codec, the method comprising:
   pre-processing control maps that were generated during encoding of the video data; and
   generating intermediate control maps comprising information regarding decoding the video data on a frame basis such that a deblocking operation is performed on an entire frame at one time, and further regarding rearranging the video data to be processed in parallel on multiple pipelines of a graphics processing unit (GPU) so as to optimize the use of the multiple pipelines.

218. The method of claim 217, further comprising executing a plurality of setup passes on the control maps, comprising performing Z-testing of a Z-buffer created from the control maps.

219. The method of claim 218, further comprising:
   determining from the intermediate control maps video data units that do not have inter-unit dependencies for deblocking filtering; and
   rearranging the video data units that do not have inter-unit dependencies such that the data units that do not have inter-unit dependencies can be processed in parallel on the multiple pipelines.

220. The method of claim 219, further comprising mapping the rearranged data units that do not have inter-unit dependencies to a scratch buffer for processing.

221. A computer readable medium including instructions which when executed in a video processing system cause the system to process the encoded video data, the processing comprising:
   pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information; and
decoded the encoded video data, wherein decoding comprises:
parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines; and
performing deblocking on a frame of video data on motion compensation has been performed.

222. The computer readable medium of claim 221, wherein the pre-defined format comprises a compression scheme according to which the video data may be encoded using one of a plurality of prediction operations for various units of data in a frame, and wherein the control information comprises an indication of which prediction operation was used to encode each unit of data in the frame.

223. The computer readable medium of claim 222, wherein the processing further comprises deblocking the decoded video data on a frame deblocking is performed on an entire frame of video data at a time.

224. The computer readable medium of claim 221, wherein the control information comprises a rearrangement of the video data such that a deblocking operation can be performed in parallel on multiple video data using the plurality of GPU pipelines.

225. The computer readable medium of claim 221, wherein pre-processing further comprises creating a Z-buffer from the control maps using one of a plurality of pre-shaders, wherein running a pre-shader on the control maps is more efficient than running a rendering shader on the control maps.

226. The computer readable medium of claim 222, wherein the compression scheme comprises one of a plurality of high-compression-ratio schemes, including H.264.

227. The computer readable medium of claim 222, wherein the pre-defined format comprises an MPEG standard video format.
228. A computer readable medium having instructions stored thereon which, when processed, are adapted to create a circuit capable of performing a method comprising:

pre-processing control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises generating a plurality of intermediate control maps containing control information, including control information specific to an architecture of a video processing unit; and decoding the encoded video data, grouping units of video data that have similar deblocking dependencies; and performing deblocking on each group having the same dependencies concurrently.

229. A computer having instructions store thereon which, when implemented in a video processing driver, cause the driver to perform a parallel processing method, the method comprising:

pre-processing control maps that were generated from encoded video data; and generating intermediate control maps comprising information regarding decoding the video data on a frame basis such that each of multiple, distinct decoding operations, including a deblocking operation, is performed on an entire frame at one time, and further regarding rearranging the video data to be processed in parallel on multiple pipelines of a graphics processing unit (GPU) so as to optimize the use of the multiple pipelines.

230. A graphics processing unit (GPU) configured to:

pre-process control maps that were generated from encoded video data; generate intermediate control maps; and use the intermediate control maps to perform deblocking of the video data on a frame basis such that deblocking is performed on an entire frame at one time, and to further rearrange the video data to be processed in parallel in groups of like dependencies on multiple pipelines of the GPU so as to optimize the use of the multiple pipelines.

231. A video processing apparatus comprising:
circuitry configured to pre-process control maps that were generated from
encoded video data that was encoded according to a predefined format, and to generate
intermediate control maps; and
driver circuitry configured to read the intermediate control maps for controlling
a video data decoding operation, and
multiple video processing pipeline circuitry configured to respond to the driver
circuitry to perform decoding of the video data on a frame basis such deblocking is
performed on an entire frame at one time, and to further rearrange the video data to be
processed in parallel in groups of like dependencies on multiple pipelines of the GPU
so as to optimize the use of the multiple pipelines.

232 A digital image generated by the method of claim 201.

233. A method for decoding video data, comprising:
a first processor generating control maps from encoded video data;
a second processor,
receiving the control maps;
generating intermediate control maps from the control maps, wherein the
intermediate control maps include information specific to an architecture of the second
processor;
using the intermediate control maps to decode the encoded video data;
deblocking the decoded data, comprising deblocking an entire frame in
parallel.

234 The method of claim 233, wherein the control maps comprise data and
control information according to a specified format.

235 The method of claim 233, further comprising the second processor using
the intermediate control maps to perform parallel processing on the video data to
generate display data

236. The method of claim 233, wherein control maps are generated on a per
frame basis.
237. The method of claim 233, wherein the architecture of the second processor comprises a type of architecture selected from a group comprising a single instruction multiple data (SIMD) architecture, a multi-core architecture; and a multi-pipeline architecture.

238. The method of claim 235, wherein parallel processing comprises performing set up passes.

239. The method of claim 238, wherein performing setup passes comprises at least one of:
   - sorting passes to sort surfaces;
   - inter-prediction passes;
   - intra-prediction passes; and
   - deblocking passes

240. A method of upgrading a system to allow for decoding of video data comprising:
   - causing an updated driver to be installed on the system, the updated driver containing computer readable instructions for adapting a system to pre-process control maps generated from encoded video data that was encoded according to a pre-defined format, wherein pre-processing comprises:
     - generating a plurality of intermediate control maps containing control information, and
     - grouping units of data with similar deblocking dependencies such that a deblocking operation is performed on units in a group concurrently

241. The method of claim 240, wherein the computer readable instructions further adapt the system to decode the encoded video data, wherein decoding comprises parallel processing using the intermediate control maps to optimize usage of a plurality of processing pipelines.

242. A hardware-accelerated decoding method, comprising:
pie-processing encoded data, wherein the encoded data is encoded in a plurality of units of predefined sizes, wherein various units of the plurality of units have dependencies, including deblocking dependencies, such that dependent units must be processed in a particular order, and wherein pre-processing comprises determining the dependencies; and

performing deblocking on all of the units in a frame in one operation.

243 The method of claim 242, wherein pre-processing further comprises designating units of data that have similar dependencies similarly,

mapping units with similar dependencies to be processed together so as to optimally utilize the hardware; and

processing similarly designated units in parallel.

244 The method of claim 243, wherein the method further comprise:
copying the mapped units to a buffer for processing; and

copying the mapped units back to the frame after processing.
FIG. 1
Control Maps

GPU

Driver

Layered Decoder

Pipe

Pipe

Pipe

Pipe

Z-buffer

Reference Buffer

To Display

FIG.2
406 Control Maps

408 Set Value to "Inter"

410 Pre-shader → Z-buffer

414 Inter Shader

416 Frame with completed Inter-prediction

418 Set Value to "Intra"

420 Pre-shader

422 Intermediate Control Maps

424 Intra Shader

426 Frame with completed Inter-prediction and Intra-prediction

To Deblocking

FIG. 4
Shader parses control map and broadcasts preprocessed information to each 4 x 4 block

Find reference frame

Find reference pels inside reference frame

Combine reference pel data and residual data

Write result to partially decoded frame

To Intra-prediction

FIG.5
Parse the control map macroblock header to determine types of subblocks within a macroblock

Assign subblocks to be rendered in the same physical pass with the same number "X"

To avoid interdependencies between the macro blocks, organize primitives (4x4 blocks) of the frame to be rendered in the same pass into a list in a diagonal fashion

Run shader on all primitives #X in parallel as allowed by HW

Is #X last number?

No

Yes

To Deblocking

FIG. 8
First Vertical pass

Writing to *3* Render Targets
Each RT represents a side of the edge

FIG.14A

FIG.14B
Filter 1 edges into 4 render targets

- Rendering 8 pixels, top 4 is top half of edge bottom 4 is bottom half of edge
FIG. 16A

Writing to *4* Render Targets
Each RT represents a side of the edge

FIG. 16B
Writing to *4* Render Targets
Each RT represents a side of the edge
Writing to *4* Render Targets
Each RT represents a side of the edge

FIG. 19B

FIG. 19A
FIG. 24

FIG. 25