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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 January 2010 (07.01.2010)

PCT

(10) International Publication Number
WO 2010/000693 A2

(51) International Patent Classification:
*H04N 7/26* (2006.01)

(21) International Application Number:
PCT/EP2009/058104

(22) International Filing Date:
29 June 2009 (29.06.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
200810131789.9 30 June 2008 (30.06.2008) CN

(71) Applicant *(for all designated States except US)*: **INTERNATIONAL BUSINESS MACHINES CORPORATION** [US/US]; New Orchard Road, Armonk, New York 10504 (US).

(71) Applicant *(for MG only)*: **IBM UNITED KINGDOM LIMITED** [GB/GB]; PO Box 41, North Harbour, Portsmouth Hampshire PO6 3AU (GB).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **YAN, Rong** [CN/CN]; IBM CHINA, MP D1B, A#1-3/f, B#1-3/F, C#1-3/F, Diamond, Zhong Guang Cun Software Park, Beijing 11 100193 (CN). **YUAN, Yu** [CN/CN]; IBM CHINA, MP D1B,A#1-3/F, B#1-3/F, C#1-3/F, Diamond, Zhong Guang Cun Software Park, Beijing 11 100193 (CN). **XU, Sheng** [CN/CN]; IBM CHINA, MP E9R, 2/F,Building 10, 399 Keyuan Road, Zhangjiang Hi-Tech Park, Pudong New District, Shanghai 201203 (CN). **YANG, Yu Dong** [CN/CN]; IBM CHINA, MP D1B, A#1-3/F, B#1-3/F, C#1-3/F, Diamond, Zhong Guang Cun Software Park, Beijing 11 100193 (CN). **LIU, Xing** [CN/CN]; IBM CHINA, Mp E8I, 399 Keyuan Rd, Zhangjiang Chuangxin, No. 10 Building, Zhangjiang High Tech Park, Shanghai 31 201203 (CN). **LI, Huo Ding** [CN/CN]; IBM CHINA, MP E9R, #28, Zhong Guan Cun Software Park, No.8 Dong Bei Wang West Road, Haidian District Beijing BJ 100193 (CN). **SHAO, Ling** [CN/CN]; IBM CHINA, MP D1B, A#1-3/F, B#1-3/F, C#1-3/F, Diamond, Zhong Guang Cun Software Park, Haidian District, Beijing 11 100193 (CN).

(74) Agent: **WALDNER, Philip**; IBM United Kingdom Limited, Intellectual Property Law, Hursley Park, Winchester Hampshire SO21 2JN (GB).

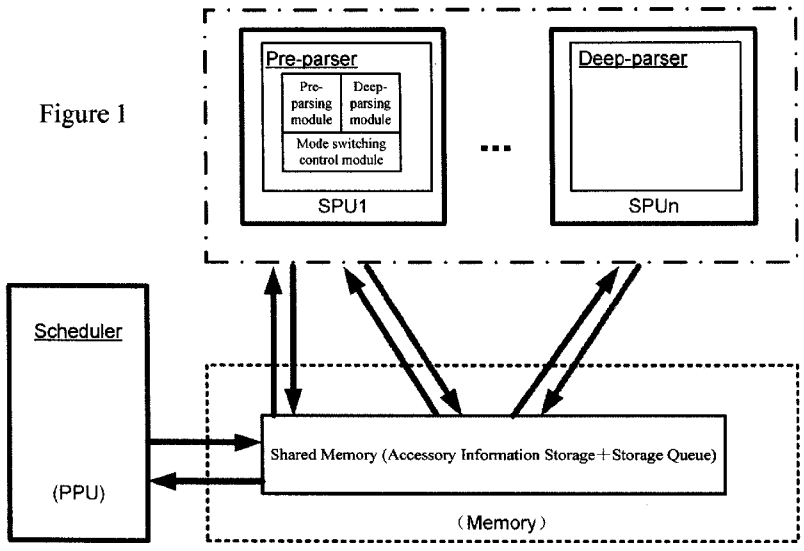
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG,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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH,

*[Continued on next page]*

(54) Title: METHOD, SYSTEM AND APPARATUS FOR PARSING MULTILAYER DATA STREAM

Figure 1




(57) Abstract: The present invention provides a method, a system and an apparatus for parsing a multilayer data stream. According to the technical solution provided by the present invention, a scheduler allocates and initializes a pre-parser, a deep-parser and a shared memory; the pre- parser pre-parse a frame in the multilayer data stream into data slices and put the pre-parsing result to the shared memory; the deep-parser gets one of the data slices from the shared memory and deep-parse the data slice into data macro-blocks. The case will not exist that the latter frame is parsed before the former frame. It is also optimized as to the storage capacity requirement and complexity of data transmission. In addition, the variability of the number of the parser makes it possible to improve the use efficiency of the parsers while keeping the parsing speed.

WO 2010/000693 A2

GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, 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, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, 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 (Rule 48.2(g))*

# METHOD, SYSTEM AND APPARATUS FOR PARSING MULTILAYER DATA STREAM

## FIELD OF THE INVENTION

The present invention relates to a field of data processing, and in particular, to a method, a system and an apparatus for parsing a multilayer data stream.

## DESCRIPTION OF THE RELATED ART

When decoding a multilayer data stream, it should firstly be parsed before it can be decoded. A typical multilayer data stream is an H.264 video data stream, wherein the data stream has one or more data frames and each data frame has one or more data slices, and each data slice has one or more data macro-blocks (MBs). When decoding the H.264 video data stream, the data frames need to be parsed into MBs before subsequent decoding can be performed. In some circumstances such as the case of mobile terminals, the processing unit has limited capabilities, therefore a single Primary Processing Unit (PPU) in the mobile terminal is not able to parse data stream with high bit rate in real time. One or more Synergistic Processing Unit (SPU) may be employed to parse the data stream parallelly.

A simple solution is to parallelly parse the data steam on a frame basis. In other words, PPU allocates different SPUs respectively to parse different data frames. In general, there is dependency between two subsequent frames in the data stream, i.e. the decoding of the latter frame depends on the decoding result of the former frame. Due to the difference between frames, the time needed for parsing frames differs. If a latter frame has been parsed by an SPU while the former frame has not been parsed by another SPU, the latter frame could not be decoded because the former frame has not been decoded. The decoding efficiency is therefore low; meanwhile the latter frame parsed needs to be buffered thus more storage capacity is required. In addition, if the storage capacity of the SPU is not capable of storing a whole frame, the PPU divides the frame into several sub-frames, sends each of the sub-frames to the SPU. When there are a large number of SPUs, the data scheduling and transmitting between the PPU and the SPUs will be very complicated.

## SUMMARY OF THE INVENTION

The present invention provides a method, a system and an apparatus for parsing a multilayer data stream. In the multilayer data stream, a data frame comprises a plurality of data slices and a data slice comprises a plurality of data macro-blocks.

According to the an aspect of the present invention, a scheduler allocates and initializes a pre-parser, a deep-parser and a shared memory; the pre-parser pre-parses frame(s) into data slices and stores the pre-parsing results to the shared memory; and the deep-parser gets one of the data slices from the shared memory and deep-parses the data slice into data macro-blocks.

The parsing is parallelly performed at the level of data slices in a same frame. A former frame is always parsed before its latter frame, therefore the case that the latter frame is parsed before the former frame will not happen. Because only one frame is buffered, the storage capacity needed is significant decreased comparing with frame-level parallel processing. The storage capacity of the SPU usually is enough for storing a data slice, therefore the data scheduling and transmission between the scheduler and the parsers as well as between the pre-parser and the deep-parser is very simple.

According to a further embodiment of the present invention, the number of the parsers is variable. In one case, the scheduler may adjust the number of the parsers at the beginning of parsing a data frame; the scheduler may also dynamically increase or decrease the number of the parsers during the process of parsing the data frame.

In another case, the number of the parsers is kept unchanged, while the pre-parser may switch between a pre-parsing mode and a deep-parsing mode. The pre-parser switches from the pre-parsing mode to the deep-parsing mode so that the number of the deep-parser is increased. In particular, the pre-parser may switch from the pre-parsing mode to the deep-parsing mode after the pre-parsing is finished. In addition, the pre-parser may, according to the data amount in the shared memory, switch from the pre-parsing mode to the deep-parsing

mode to prevent the shared memory from overflow, or switch from the deep-parsing mode to the pre-parsing mode to prevent the deep-parser from being idle.

The variability of the number of the parser makes it possible to improve the efficiency of the parsers usage while keeping the parsing speed.

## BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram illustrating the system for parsing a multilayer data stream according to an embodiment of the present invention.

Figure 2 is a schematic diagram illustrating a shared memory according to an embodiment of the present invention.

Figure 3 is an overall flowchart of the method of parsing a multilayer data stream according to an embodiment of the present invention.

Figure 4 is a detailed flowchart of deep parsing data slices according to an embodiment of the present invention.

Figure 5 is an example of parsing the multilayer data stream according to an embodiment of the present invention.

## DETAILED DESCRIPTION OF THE EMBODIMENTS

The method, system and apparatus for parsing multilayer data stream according to the present invention will be described in detail with reference to the accompanying drawings. It is understood that, terms such as data frames, data slices and data MBs used in the specification are only names of respective layers in a multilayer data stream. When the present invention is applied to different kinds of data streams, there may be different names.

Figure 1 is a block diagram illustrating the system for parsing a multilayer data stream according to an embodiment of the present invention.

As shown in Figure 1, the system for parsing a multilayer data stream according to an embodiment of the present invention comprises: a scheduler, a pre-parser, a deep parser and a shared memory. The pre-parser and the deep-parser may be referred to in general as a parser. The shared memory comprises auxiliary information storage and a store queue.

The scheduler is usually located in a PPU, and is configured to allocate and initialize a pre-parser and a deep-parser for parsing. The scheduler is also configured to allocate and initialize the shared memory. The parsers are mapped to respective SPUs.

The scheduler schedules the number of the parsers on a frame basic. In other words, the scheduler determines the number of the parsers according to the time needed to parse the previous frame of the same type, the number of the parsers used to parse the previous frame of the same type and a real-time parsing time limitation. For example, if the scheduler finds out that the time needed to parse the previous frame of the same type is longer than the real-time parsing time limitation, the scheduler may increase the number of the parsers used to parse the current frame; on the contrary, if the scheduler finds out that the time needed to parse the previous frame of the same type is shorter than the real-time parsing time limitation, the scheduler may decrease the number of the parsers used to parse the current frame. The above mentioned real-time parsing time limitation is a time length reference to parse a certain type of frames in order to parse the whole video stream in real time. The real-time parsing time limitation may be a specific value or an interval. The real-time parsing time limitations for different types of frames may vary. For example, in the case of H.264, the real-time parsing time limitation for an I frame is generally greater than that for a P frame, and the real-time parsing time limitation for a P frame is generally greater than that for a B frame. In the case that all frames in the multilayer data frame are of the same type, the previous frame of the same type is exactly the previous frame, and all data frames have the same real-time parsing time limitation. The scheduler may include a semi-dynamically scheduling module to perform the scheduling on the frame basic.

In addition, the scheduler may dynamically increase or decrease the number of the parsers during the process of parsing the current frame. For example, the PPU may calculate a parsing rate of the current frame during the process of parsing it, such as the rate of the number of the parsed data MBs of the current frame versus the number of the total data MBs of the current frame; the PPU may also calculate a time rate of the time needed so far for parsing the current frame versus the real-time parsing time limitation. If the parsing rate is greater than the time rate, i.e. the parsing rate precedes, the PPU decrease the number of the parsers; if the parsing rate is smaller than the time rate, i.e. the parsing rate lags, the PPU increase the number of the parsers. The calculation and the scheduling may be performed at a certain frequency, or be trigged by some events, for example each time when the parsers put data to the store queue or when the parsers get data from the store queue. There may be other dynamic scheduling algorithms. The dynamic scheduling is able to improve the efficiency of the parsers, however at the expense of more calculation costs. The scheduler may include a dynamic scheduling module to perform the dynamic scheduling.

If the efficiency of the parsers usage is not a problem, the PPU may allocate and initialize the parsers only at the beginning of parsing the multilayer data stream and the number of the parsers may be kept unchanged through the process of parsing the whole multilayer data stream. This will save the calculation costs needed for scheduling.

Among the parsers, at least one parser is initialized as the pre-parser, which is capable of switching between a pre-parsing mode and a deep-parsing mode and is of the pre-parsing mode at the beginning of parsing a frame. In other words, the pre-parser includes a pre-parsing module, a deep-parsing module and a mode switching control module. The other parsers are initialized as deep-parsers.

The pre-parser pre-parses the data frame into data slices, and put the data slices into the store queue in order. The deep-parser each time gets one data slice from the head of the store queue, deep-parses the data slice into data MBs, sends the data MBs to PPU for subsequent processing, and get another data slice. After all data slices of one frame are put to the store queue, the mode switching control module switches the pre-parser from the pre-parsing

mode to the deep-parsing mode, i.e. stops the pre-parsing module and starts the deep-parsing module.

In practice, in the case that the capability of the store queue is not enough for all data slices in one frame, there may be mismatch between the speed at which the pre-parser put data slices to the queue and the speed at which the deep-parser get data slices from the queue. If it is long enough that the former speed is greater than the latter speed, there may be overflow; if it is long enough that the former speed is less than the latter speed, there may underflow. To solve the problem, the mode switching control module further monitors the data amount in the store queue. When the data amount in the store queue exceeds a first threshold, the mode switching control module switches the pre-parser from the pre-parsing mode to the deep-parsing mode, i.e. stops the pre-parsing module and starts the deep-parsing module, so that the pre-parser operates as a deep-parser. When the data amount in the store queue is below a second threshold, the mode switching control module switches the pre-parser from the deep-parsing mode to the pre-parsing mode, i.e. stops the deep-parsing module and starts the pre-parsing module, so that the pre-parser continues to parse the frame into data slices and put the data slices to the store queue. The above mentioned threshold may be an absolute data amount, or a percentage of the data amount of the data slices currently in the store queue versus the capacity of the store queue. The first threshold may be the same as the second threshold. In a preferred embodiment, the first threshold is greater than the second threshold to prevent frequent switching of the pre-parser.

It is understood that, the mode switching control module and the deep-parsing module are not necessary modules of the pre-parser. When the number of the parser is sufficient enough, there is no need to switch the pre-parser from the pre-parsing mode to the deep-parsing mode.

The store queue is configured to store the data slices obtained by the pre-parser. According to the definition of a queue, the data slices are First-In-First-Out in the store queue. In other words, the pre-parser puts the obtained data slices to the end of the queue, and the deep-parser gets the data slices from the head of the queue. The accessory information storage

stores therein accessory information. An example of the shared memory is shown in Figure 2, which will be described in detail below.

Figure 3 is an overall flowchart of the method of parsing a multilayer data stream according to an embodiment of the present invention.

Step 301, at the beginning of a frame, the scheduler allocates and initializes parsers for parsing the current frame. The scheduler also allocates and initializes the shared memory. As mentioned above, the number of the parsers is determined at the beginning of the frame according to the time needed to parse the previous frame of the same type, the number of the parsers used to parse the previous frame of the same type and the real-time parsing time limitation. The number of the parsers may also be adjusted dynamically during the parsing process. The number of the parsers may also be determined when the process of parsing the whole multilayer data stream begins, and be kept unchanged thought the process.

Among the parsers, at least one parser is initialized as the pre-parser, which is capable of switching between a pre-parsing mode and a deep-parsing mode and is of the pre-parsing mode at the beginning of parsing the frame. The other parsers are initialized as the deep-parser.

The PPU also allocates and initializes the shared memory. In particular, the PPU initializes the auxiliary information in the shared memory. The detailed initialization will be described below.

Step 302, the pre-parser parses the data frame to data slices and put the data slices to the store queue. The data slices are shown in Figure 2 as Data Slice 1, Data Slice 2, and etc. As mentioned above, there is some auxiliary information in the auxiliary information storage. The auxiliary information may include the identity of the current frame (FRAME NUMBER) and the total number of the data slices in the current frame (MAX SLICE) and etc. FRAME NUMBER may be set by the PPU or the pre-parser. If the header of the frame records the total number of the data slices in the frame, the PPU or the pre-parser initializes MAX SLICE as the value recorded in the header. If the header of the frame records the

length of the frame indicated by the number of the bits in the frame, the PPU or the pre-parser initializes MAX SLICE as zero, afterwards the pre-parser increases MAX SLICE by 1 each time when the pre-parser obtains one data slice. In other words, according to the definition of the frame structure, MAX SLICE may be kept unchanged during the process of parsing the frame, or be increased with the process.

Step 303, in the case that the capability of the store queue is not enough for all data slices in one data frame, the deep-parser determines, according to the data amount in the store queue, whether to switch from the pre-parsing mode to the deep-parsing mode temporarily, or whether to switch back. The step may be omitted.

Step 304, the pre-parser determines whether the pre-parsing has finished, i.e. whether all data slices in the data frame has been obtained by parsing. If the pre-parsing has finished, the pre-parser switches to the deep-parsing mode and operates as a deep-parser. The pre-parser may determine whether all data slices in the data frame has been obtained according to the total number of the data slices or the number of the bits recorded in the header of the data frame.

Step 305, it is determined whether the current frame has been parsed. If the current frame has been parsed, proceed to step 307 in which the PPU is informed to get all data MBs; otherwise, proceed to step 306.

Step 306, the deep-parser or the pre-parser having switched to the deep-parsing mode get one data slice from the store queue, and deep-parses the data slice. Data MBs obtained through the deep-parsing are put to buffers specified by the PPU.

The detailed operations of a deep-parser in Step 305 and Step 306 will now be described with reference to Figure 4. The pre-parser having switched to the deep-parsing mode operates the same as the deep-parser with respect to Step 305 and Step 306.

Step 3051, the deep-parser read from the accessory information storage FRAME NUMBER、MAX SLICE、LAST FETCHED SLICE、CUMULATIVE MB and SLICE

DATA OFFSET. FRAME NUMBER and MAX SLICE are set by the pre-parser. The other accessory information is set by the deep-parser after finishing deep-parsing one data slice. LAST FETCHED SLICE indicates how many data slices have been parsed as to the current frame, and is initialized to zero in Step 301; CUMULATIVE MB indicates how many MBs have been obtained as to the current frame, and is initialized to zero in Step 301; SLICE DATA OFFSET indicates the head of the store queue, and is initialized to the start point of the queue in Step 301.

Step 3052, the deep-parser determines whether the following condition is true: CUMULATIVE MB equals MAX MB and LAST FETCHED SLICE equals MAX SLICE. If the condition is true, proceed to Step 307, otherwise proceed to Step 3061. MAX MB indicates the max number of the MBs as to the current frame according to the definition of the frame structure.

Step 3061, the deep-parser gets one data slice from the store queue according to the value of SLICE DATA OFFSET and deep-parses the data slice.

Step 3062, the deep-parser modifies the value of LAST FETCHED SLICE、CUMULATIVE MB and SLICE DATA OFFSET. LAST FETCHED SLICE is increased by 1, CUMULATIVE M is increased by the number of the MBs obtained by deep-parsing the current data slice, and SLICE DATA OFFSE is set to the position of the next data slice in the store queue. LAST FETCHED SLICE and SLICE DATA OFFSET may alternatively be modified before deep-parsing the current data slice, i.e. before Step 3061.

An example will be described with reference to Figure 5 in which the horizontal axis indicates the time. In this example, a frame 0 is parsed using two parsers. For the sake of clear description, among the accessory information LAST FETCHED SLICE、CUMULATIVE MB and SLICE DATA OFFSET, only the changes in LAST FETCHED SLICE and CUMULATIVE MB will be described. In addition, the deep-parser modifies LAST FETCHED SLICE before the deep-parsing begins, and modifies CUMULATIVE MB finishes. It is further assumed that the current frame has 12 data slices.

The pre-parser is SPU0, which set FRAME NUMBER to zero. In this example, it is assumed that the header of the data frame does not record the total number of the data slices in the frame, therefore SPU0 further initializes MAX SLICE as zero. According to Step 302, SPU0 sets MAX SLICE to 1 after the first data slice, i.e. slice 1, is obtained by pre-parsing the data frame.

At time point A, the deep-parser SPU1 begins deep-parsing slice 1. The deep-parser SPU1 modifies LAST FETCHED SLICE from its initial value 0 to 1. CUMULATIVE MB is of its initial value 0.

At time point B, deep-parser SPU1 finishes deep-parsing slice 1, and modifies CUMULATIVE MB to 60. Here, 60 is the number of the data MBs obtained from deep-parsing slice 1. Also at time point B, deep-parser SPU1 begins deep-parsing slice 2, and modifies LAST FETCHED SLICE to 2. In addition, during the interval between time point A and time point B, the pre-parser obtains slice 2 through slice 6 by pre-parsing the frame. Therefore MAX SLICE is 6 at time point B.

At time point C, the deep-parser SPU1 does not finish deep-parsing slice2 and CUMULATIVE MB is still 60. In addition, during the interval between time point B and time point C, the pre-parser SPU0 obtains slice7 through slice 12. Therefore MAX SLICE is 12 at time point C. According to the assumption, the current frame has 12 data slices, SPU0 switches to the deep-parsing mode, begins deep-parsing slice3 and modifies LAST FETCHED SLICE to 3, as described in Step 302.

At time point D, the deep-parser SPU1 finishes deep-parsing slice 2, and modifies CUMULATIVE MB to 120; in other words, there are 60 data MBs obtained by deep-parsing slice 2. Also, the deep-parser SPU1 begins deep-parsing slice, and modifies LAST FETCHED SLICE to 4. From time point D on, no pre-parsing is performed, therefore MAX SLICE is kept 12.

Thereafter, the pre-parser SPU0 which is in the deep-parsing mode and the deep-parser SPU1 deep-parse the other data slices, and check whether CUMULATIVE MB reaches the

max number 8160 of the MBs as to the current frame according to the definition of the frame structure.

At time point J, the deep-parser SPU1 finds that CUMULATIVE MB reaches the max number 8160 of the MBs as to the current frame according to the definition of the frame structure, and determines that the current frame has been parsed, so that the deep-parser SPU1 informs PPU to perform subsequent processing.

Although specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that modifications may be made on these embodiments without departing from the principle and spirit of the present invention. The scope of the present invention is determined by the claims and their equivalence.

## CLAIMS

1. A method for parsing a multilayer data stream, wherein a data frame in the multiplayer data stream comprises a plurality of data slices and a data slice comprises a plurality of data macro-blocks, the method, during a process of parsing data frames, comprising:

allocating and initializing, by a scheduler, a pre-parser, a deep-parser and a shared memory;

pre-parsing, by the pre-parser, the data frame into data slices, and putting pre-parsing results to the shared memory; and

getting, by the deep-parser, one of the data slices from the shared memory, and deep-parsing the data slice into data macro-blocks.

2. The method according to Claim 1, wherein the scheduler allocating and initializing the pre-parser and the deep-parser comprises:

determining, by the scheduler, the number of the parsers at the beginning of parsing the data frame; and

allocating and initializing, by the scheduler, the pre-parser and the deep-parser according to the determined number.

3. The method according to Claim 2, wherein determining the number of the parsers comprises:

if the time needed to parse a previous data frame is longer than a time reference, the number of the parsers for parsing the current parser is increased; if the time needed to parse a previous data frame is shorter than the time reference, the number of the parsers for parsing the current parser is decreased.

4. The method according to Claim 3, wherein data frames in the multilayer data stream are of different types, the time needed to parse the previous data frame is the time needed to parse the previous data frame of the same type, and the time reference is the time reference corresponding to the type.

5. The method according any one of Claims 1-4, further comprising:

increasing or decreasing dynamically the number of the parsers during the process of parsing the data frame.

6. The method according to Claim 5, wherein increasing or decreasing dynamically the number of the parsers during the process of parsing the data frame comprises:

calculating a parsing rate of the current data frame; and

increasing the number of the parsers if the parsing rate lags, or decreasing the number of the parsers if the parsing rate precedes.

7. The method according to any one of Claims 1-6, further comprising:

the pre-parser switching from a pre-parsing mode to a deep-parsing mode if all data slices in the data frame have been obtained by the pre-parser.

8. The method according to any one of Claims 1-7, further comprising:

the pre-parser switching between the pre-parsing mode and the deep-parsing mode according to data amount in the shared memory.

9. The method according to Claim 8, wherein the pre-parser switching between the pre-parsing mode and the deep-parsing mode according to data amount in the shared memory comprises:

the pre-parser switching from the pre-parsing mode to the deep-parsing mode if the data amount in the shared memory is over a first threshold; and

the pre-parser switching from the deep-parsing mode to the pre-parsing mode if the data amount in the shared memory is under a second threshold.

10. A method for parsing a multilayer data stream, wherein a data frame in the multiplayer data stream comprises a plurality of data slices and a data slice comprises a plurality of data macro-blocks, wherein a pre-parser pre-parses a current data frame into data slices and switches from a pre-parsing mode to a deep-parsing mode if all data slices in the current data frame have been obtained by the pre-parser.

11. The method according to Claim 10, further comprising:

the pre-parser putting pre-parsing results to a shared memory, and switching between the pre-parsing mode and the deep-parsing mode according to data amount in the shared memory.

12. The method according to Claim 11, wherein the pre-parser switching between the pre-parsing mode and the deep-parsing mode according to data amount in the shared memory comprises:

the pre-parser switching from the pre-parsing mode to the deep-parsing mode if the data amount in the shared memory is over a first threshold; and

the pre-parser switching from the deep-parsing mode to the pre-parsing mode if the data amount in the shared memory is under a second threshold.

13. A system for parsing a multilayer data stream, wherein a data frame in the multiplayer data stream comprises a plurality of data slices and a data slice comprises a plurality of data macro-blocks, comprising:

a scheduler configured to allocate and initialize a parser and a shared memory, wherein the parser comprises a pre-parser and a deep-parser;

the pre-parser, comprising a pre-parsing module which is configured to pre-parse the data frame into data slices and put the pre-parsing result to the shared memory;

the deep-parser configured to get one of the data slices from the shared memory and deep-parse the data slice into data macro-blocks; and

the shared memory configured to store the data slices obtained by the pre-parser.

14. The system according to Claim 13, wherein the scheduler comprises a semi-dynamically scheduling module configured to determine the number of the parsers at the beginning of parsing the data frame, and allocate and initialized the pre-parser and the deep-parser according to the determined number.

15. A pre-parser for parsing a multilayer data stream, wherein a data frame in the multiplayer data stream comprises a plurality of data slices and a data slice comprises a plurality of data macro-blocks, comprising:

a pre-parsing module, configured to pre-parse the data frame into data slices and put the pre-parsing result to the shared memory;

a deep-parsing module, configured to get the data slice from the shared memory and parse the data slice into the data macro-blocks; and

a mode switching control module, configured to stop the pre-parsing module and starts the deep-parsing module if all data slices in the data frame have been obtained by the pre-parser.

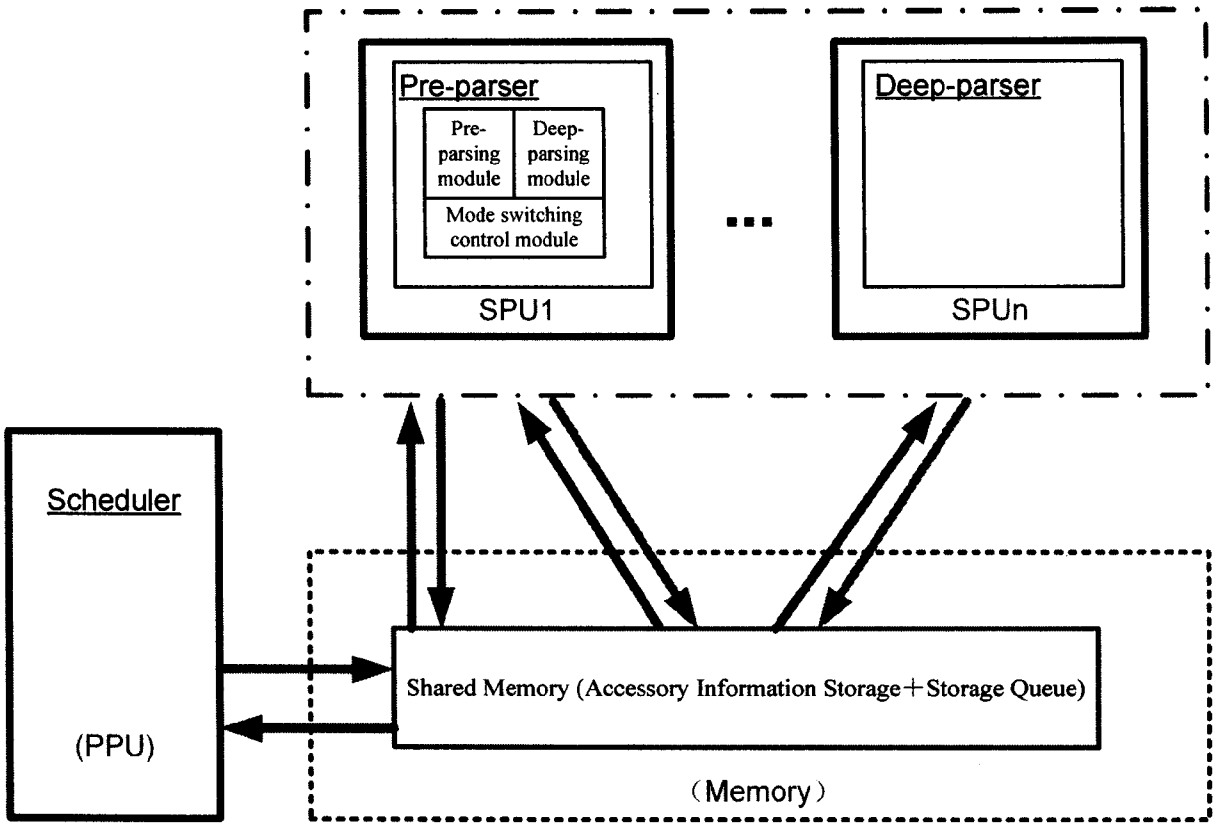

Pre-parser
Pre-parsing module
Deep-parsing module
Mode switching control module
SPU1
Deep-parser
SPUn
Scheduler
(PPU)
Shared Memory (Accessory Information Storage+Storage Queue)
(Memory)

Figure 1

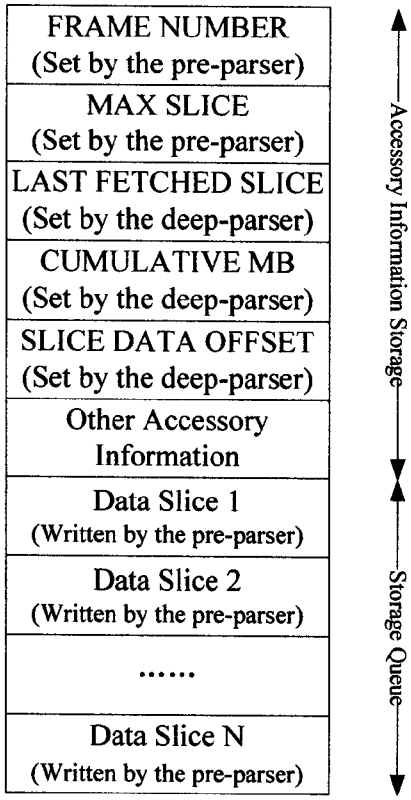

FRAME NUMBER
(Set by the pre-parser)
MAX SLICE
(Set by the pre-parser)
LAST FETCHED SLICE
(Set by the deep-parser)
CUMULATIVE MB
(Set by the deep-parser)
SLICE DATA OFFSET
(Set by the deep-parser)
Other Accessory Information
Data Slice 1
(Written by the pre-parser)
Data Slice 2
(Written by the pre-parser)
.....
Data Slice N
(Written by the pre-parser)
Accessory Information Storage
Storage Queue

Figure 2

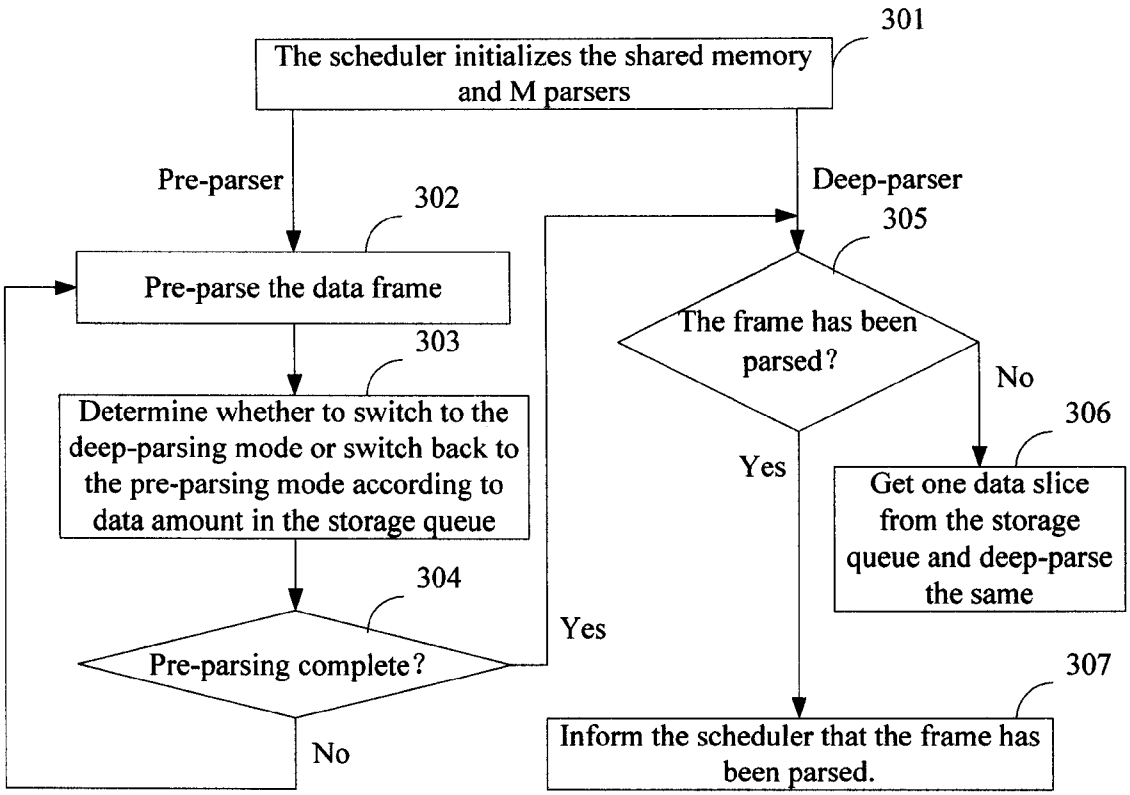

301
The scheduler initializes the shared memory and M parsers
Pre-parser
Deep-parser
302
Pre-parse the data frame
303
Determine whether to switch to the deep-parsing mode or switch back to the pre-parsing mode according to data amount in the storage queue
304
Pre-parsing complete?
Yes
No
305
The frame has been parsed?
No
Yes
306
Get one data slice from the storage queue and deep-parse the same
307
Inform the scheduler that the frame has been parsed.

Figure 3

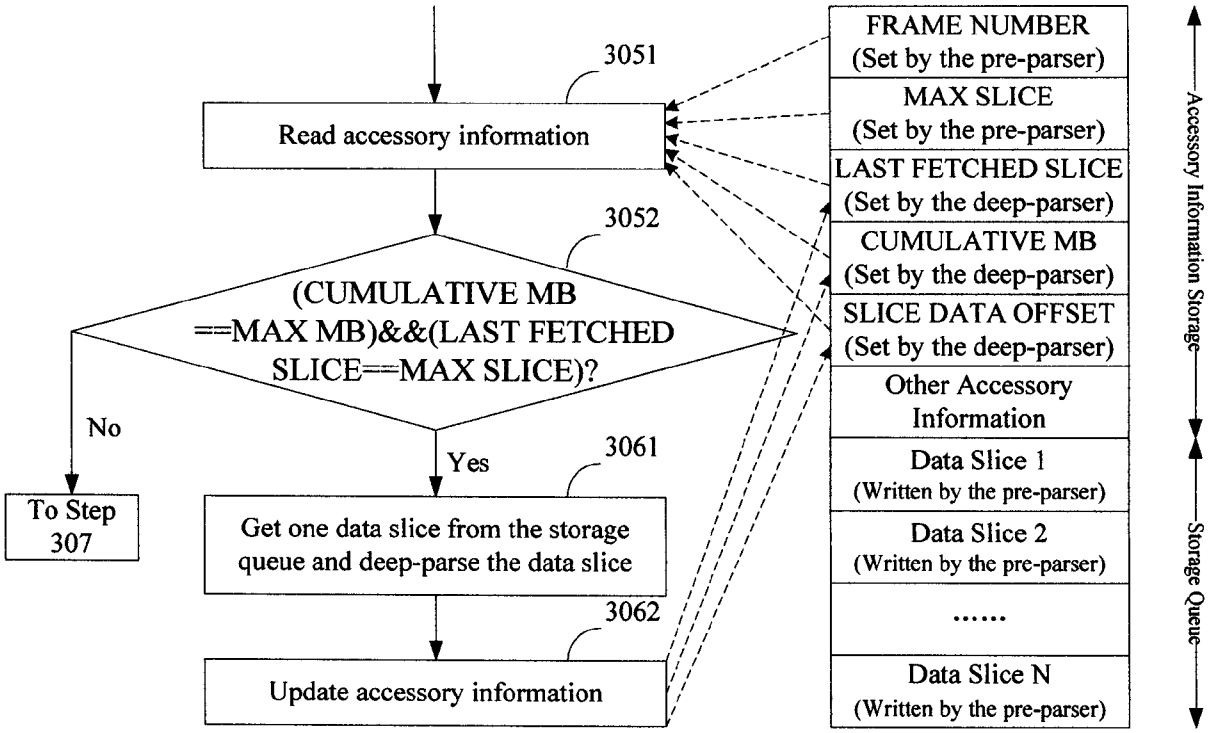

3051
Read accessory information
3052
(CUMULATIVE MB ==MAX MB)&&(LAST FETCHED SLICE==MAX SLICE)?
No
Yes
To Step 307
3061
Get one data slice from the storage queue and deep-parse the data slice
3062
Update accessory information
FRAME NUMBER (Set by the pre-parser)
MAX SLICE (Set by the pre-parser)
LAST FETCHED SLICE (Set by the deep-parser)
CUMULATIVE MB (Set by the deep-parser)
SLICE DATA OFFSET (Set by the deep-parser)
Other Accessory Information
Data Slice 1 (Written by the pre-parser)
Data Slice 2 (Written by the pre-parser)
.....
Data Slice N (Written by the pre-parser)
Accessory Information Storage
Storage Queue

Figure 4

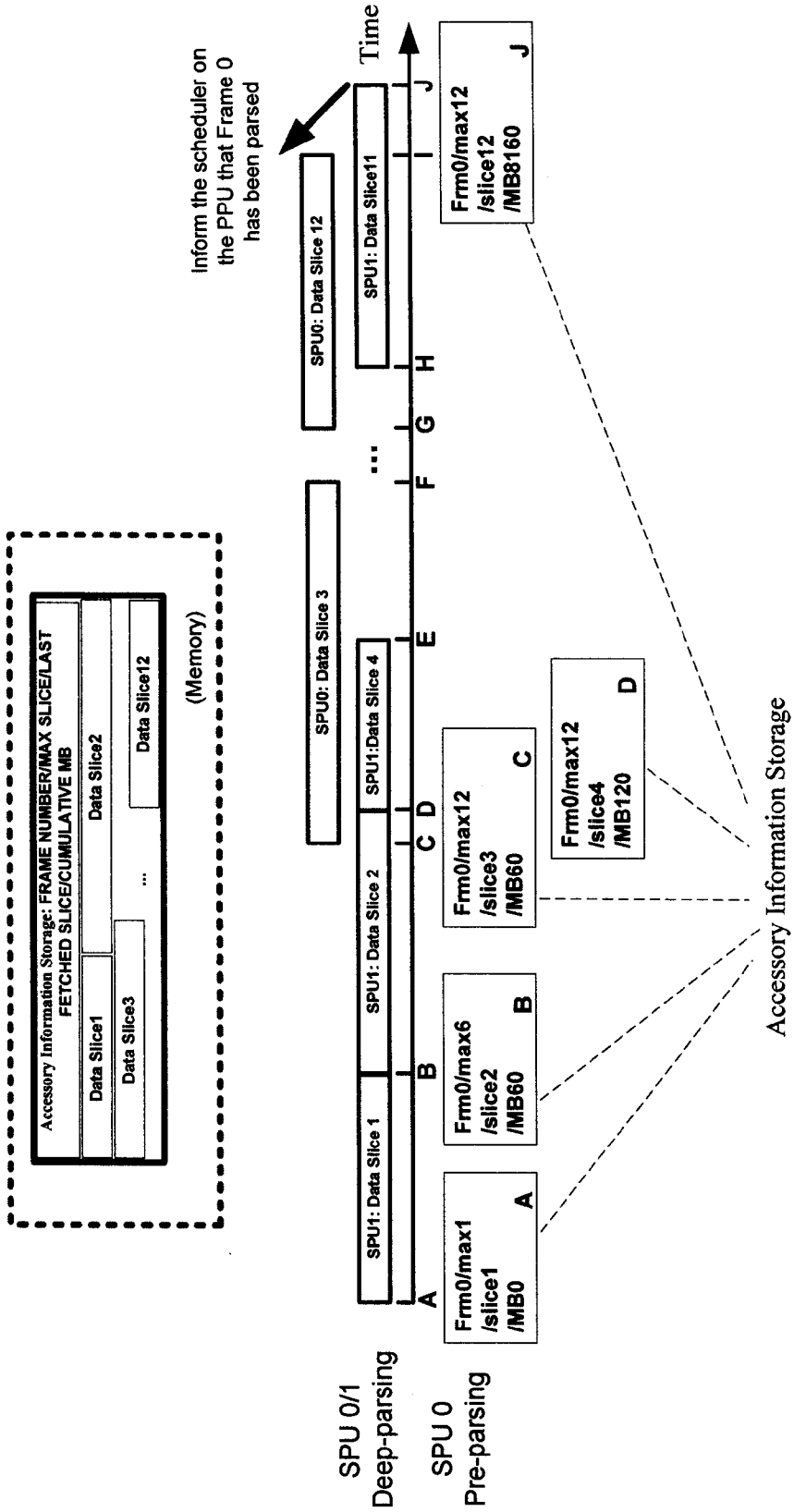

Accessory Information Storage: FRAME NUMBER/MAX SLICE/LAST FETCHED SLICE/CUMULATIVE MB
Data Slice1
Data Slice2
Data Slice3
...
Data Slice12
(Memory)
Inform the scheduler on the PPU that Frame 0 has been parsed
Time
SPU0: Data Slice 12
SPU1: Data Slice11
SPU0: Data Slice 3
SPU1:Data Slice 4
SPU1: Data Slice 2
SPU1: Data Slice 1
A
B
C
D
E
F
G
H
I
J
Frm0/max1 /slice1 /MB0 A
Frm0/max6 /slice2 /MB60 B
Frm0/max12 /slice3 /MB60 C
Frm0/max12 /slice4 /MB120 D
Frm0/max12 /slice12 /MB8160 J
SPU 0/1 Deep-parsing
SPU 0 Pre-parsing
Accessory Information Storage

Figure 5