

Memory Management Method and Apparatus, Control Program and Controller

Cross-Reference to Related Application

This disclosure is based on Chinese patent application CN 20211028812.3 filed on 02 September 2021, entitled "MEMORY MANAGEMENT METHOD AND APPARATUS, CONTROL PROGRAM AND CONTROLLER", and claims the priority of this patent application, the content of which is incorporated herein by reference in its entirety.

Technical Field

The present disclosure relates to the field of communications, and in particular, to a memory management method and apparatus, a control program, and a controller.

Background

In applications of the Ethernet switching chips, different types of off-chip memory units, i.e., memory controllers, are usually selected according to application scenarios and costs, so a Memory Management Unit (MMU) of a chip not only needs to meet basic function requirements, but also needs good compatibility and portability, so that different memory controllers can be adopted in different application scenarios according to various factors, such as storage size, speed, power consumption, and costs, they do not need to be developed individually, thus the manpower and costs are saved.

With the continuous improvement of frequency and bandwidth of processors and memory controllers, their respective performances are also continuously improved, however the memory access efficiency has become a bottleneck of system performance, and the memory access efficiency is related to an implementation manner of an MMU. In an Ethernet switching chip, the MMU is mainly responsible for the data of a packet (PK), a write request and write data distribution of a packet descriptor (PD), and order-preserving release according to the write request; then, order-preserving reading back the data according to the read request. In this process, the memory bandwidth is maximally utilized by using technologies such as off-chip address management, physical address mapping, and co-packaging, so as to improve the efficiency of the memory controller. The current mainstream memory controller includes DDR3/DDR4/DDR5 (Double Data Rate) and HBM (High Bandwidth Memory) etc. How to be compatible with different memory controllers under the same framework and ensure the storage efficiency thereof is a main problem

to be solved.

For the problem of how to be compatible with different memory controllers in the same framework, no effective solution has been proposed at present.

Summary

Embodiments of the present disclosure provide a memory management method and apparatus, a control program, and a controller, so as to at least solve a problem in the related art that how to be compatible with different controllers in a same framework.

According to an embodiment of the present disclosure, a memory management method is provided, including: identifying, by an MMU, the type of a memory controller externally connected based on CPU configuration information; determining an offset address in a manner of table look-up based on the address management sub-module and an address area corresponding to the type of the memory controller; calculating a logic address of the type of the memory controller based on the offset address, and the numbers of external connection channels gated by different types of memory controllers are different.

According to another embodiment of the present disclosure, a memory management apparatus is provided, including: an identification unit configured to enable an MMU to identify, based on CPU configuration information, the type of a memory controller externally connected; a determination unit configured to determine an offset address in a manner of table look-up based on an address management sub-module and an address area corresponding to the type of the memory controller; a calculation unit configured to calculate a logic address of the type of the memory controller based on the offset address, and the numbers of external connection channels gated by different types of memory controllers are different.

According to another embodiment of the present disclosure, a computer-readable storage control program is further provided, wherein the computer readable storage control program stores a computer program, and when the computer program is executed by a processor, the computer program is configured to perform the steps in the above embodiments.

According to another embodiment of the present disclosure, also provided is a controller, comprising a memory and a processor, wherein the memory stores a computer program, and the processor is configured to executed the computer program to perform the steps in the above embodiments.

By means of the present disclosure, an MMU is used to identify the type of a memory controller

externally connected based on CPU configuration information; based on the address management sub-module and an address area corresponding to the type of the memory controller, determining an offset address in a manner of table look-up; calculating a logic address of the type of the memory controller based on the offset address, and the numbers of external connection channels gated by different types of memory controllers are different; therefore, switching of different controllers in the same framework is achieved, the problem of being compatible with different controllers in the same framework can be solved, multiple controllers can be compatible in the same framework, and the storage efficiency is improved.

Brief Description of the Drawings

Figure 1 is a hardware structure block diagram of a mobile terminal of a memory management method according to an embodiment of the present disclosure;

Figure 2 is a flowchart of a memory management method according to an embodiment of the present disclosure;

Figure 3 is an architecture diagram of a memory management system according to an embodiment of the present disclosure;

Figure 4 is a schematic diagram of an option E mode of an HBM in a memory management method according to an embodiment of the present disclosure;

Figure 5 is a first schematic diagram of address correspondence in a memory management method according to an embodiment of the present disclosure;

Figure 6 is a second schematic diagram of address correspondence in a memory management method according to an embodiment of the present disclosure;

Figure 7 is a third schematic diagram of address correspondence in a memory management method according to an embodiment of the present disclosure;

Figure 8 is a fourth schematic diagram of address correspondence in a memory management method according to an embodiment of the present disclosure;

Figure 9 is a first schematic diagram of address mapping in a memory management method according to an embodiment of the present disclosure;

Figure 10 is a second schematic diagram of address mapping in a memory management method according to an embodiment of the present disclosure;

Figure 11 is a third schematic diagram of address mapping in a memory management method according to an embodiment of the present disclosure;

Figure 12 is a schematic diagram of off-chip address management in a memory management method according to an embodiment of the present disclosure;

Figure 13 is a schematic structural diagram of a memory management apparatus according to an embodiment of the present disclosure.

Detailed Description of the Embodiments

Embodiments of the present disclosure will be described below in detail with reference to the accompanying drawings and in conjunction with embodiments.

It should be noted that, terms such as "first" and "second" in the description, claims, and accompanying drawings of the present disclosure are used to distinguish similar objects, but are not necessarily used to describe a specific sequence or order.

The methods provided in the embodiments of the present disclosure can be executed in a mobile terminal, a computer terminal or a similar computing device. Taking the mobile terminal as an example, figure 1 is a hardware structure block diagram of a mobile terminal in a memory management method according to an embodiment of the present disclosure. As shown in figure 1, the mobile terminal may include one or more (only one is shown in figure 1) processor 102 (the processor 102 may be, but not limited to, a processing apparatus such as an MCU or an FPGA) and a memory 104 for storing data, wherein the mobile terminal can further include a transmission device 106 and an input/output device 108 for a communication function. A person of ordinary skill in the art may understand that the structure shown in figure 1 is merely exemplary, which does not limit the structure of the foregoing mobile terminal. For example, the mobile terminal may further include more or less components than shown in figure 1, or have a different configuration from that shown in figure 1.

The memory 104 may be configured to store a computer program, for example, a software program and a module of application software, such as a computer program corresponding to the memory management method in the embodiments of the present disclosure. The processor 102 runs the computer program stored in the memory 104, so as to execute various function applications and data processing, that is, to implement the foregoing method. Memory 104 may include high-speed random access memory, and may also include non-transitory memory, such as one or more magnetic storage devices, flash memory, or other non-transitory solid-state memory. In some instances, memory 104 may further include memory remotely located with respect to

processor 102, which may be connected to mobile terminals over a network. Examples of such networks include, but are not limited to, the Internet, intranets, local area networks, mobile communication networks, and combinations thereof.

The transmission device 106 is configured to receive or transmit data via a network. Specific examples of the described network may include a wireless network provided by a communication provider of the mobile terminal. In one example, the transmission device 106 may comprise a network interface controller (NIC) that may be coupled to other network devices via a base station to communicate with the Internet. In one example, the transmission device 106 can be a radio frequency (RF) module for communicating wirelessly with the Internet.

Figure 2 is a flowchart of a memory management method according to an embodiment of the present disclosure. As shown in figure 2, the flow includes the following steps:

at S202, an MMU identifies the type of a memory controller externally connected based on the configuration information of a CPU;

at S204, based on an address area corresponding to the address management sub-module and the type of the memory controller, determining an offset address in a manner of table look-up;

at S206, calculating a logic address of the type of the memory controller based on the offset address, and the numbers of external connection channels gated by different types of memory controllers are different.

According to the embodiments of the present disclosure, the MMU is configured to identify the type of a memory controller externally connected based on CPU configuration information; based on the address management sub-module and an address area corresponding to the type of the memory controller, determining an offset address in a manner of table look-up; calculating a logic address of the type of the memory controller based on the offset address, and the numbers of external connection channels gated by different types of memory controllers are different; therefore, switching of different controllers in the same framework is achieved, and the problem of being compatible with different controllers in the same framework can be solved, so that a plurality of controllers can be compatible in the same framework, and the effect of improving the storage efficiency can be achieved.

In one or more embodiments, the memory management method further comprises: an address mapping submodule of the MMU reads a preset address mapping relationship, and the address mapping relationship is for converting an on-chip logical address into a physical address that can be accepted by the memory chip.

In one or more embodiments, the memory management method further comprises: an address mapping submodule of the MMU reads a preset address mapping, and the address mapping relationship is for converting an on-chip logic address into a physical address that can be accepted by a memory chip; according to different application scenarios, performing reconfiguration by means of a CPU so as to acquire a reconfigured address mapping relationship corresponding to the application scenarios.

In one or more embodiments, the memory management method further comprises: segmenting, according to a off-chip memory requirement and a structural attribute of the memory controller, a data packet sent by a CPU to the MMU according to a block unit, and the segmented data packets corresponds to block addresses, one data packet corresponds to a plurality of block addresses, and the managed address ranges are different when types of the memory controller are different.

In one or more embodiments, the memory management method further comprises: receiving, by the MMU, off-chip data sent by a packet memory unit (PMU), and the off-chip data comprises a packet descriptor (PD) and a packet (PK); extracting data information of the PD; co-packing the data information to obtain co-packed data, and deleting the first data in the PK; and the first data may include invalid data in the co-packed data.

Performing shift and concatenations on the co-packed data according to the effective data bits, and extracting second data, and the second data may include valid data in the co-packed data. The data information of the PD in the co-packed data and the extracted second data are further packed to obtain target co-packed data. The target co-packed data is sent to the off-chip memory for storage.

In one or more embodiments, the memory management method further comprises: when the data bit length of the PD plus the data bit length of the extracted second data is less than or equal to the bus bit width, outputting the PD and the second data at the same time;

Splitting the PD and the second data into the first split data and the second split data when the data bit length of the PD plus the data bit length of the extracted second data is greater than the bus bit width;

Outputting the PD and the first split data;

Performing a zero-padding operation on the second split data to obtain zero-padding processed data with a data bit length equal to the bus bit width, and outputting the zero-padding processed data.

In one or more embodiments, the above memory management method further comprises: a packet memory unit (PMU) issuing a write packet;

The PMU stores the write packet, and sends a write release command to a Traffic Memory management unit (TMMU); the TMMU sends a packet descriptor to a queue management unit (QMU);

The QMU sends a write command via the TMMU, and transparently transmits to the write command the MMU; after finishing storing the write command, the QMU sends a write release signal to the TMMU;

The command queue issues a reading packet command, reads the data of the reading packet and returns the data of reading packet to the PMU;

The TMMU sends a read command and reads the data of the packet descriptor.

Based on the foregoing embodiments, in an application embodiment, in a memory management apparatus provided by the present disclosure, an MMU is located between an off-chip storage controller and an on-chip control module, figure 3 is an architecture diagram of a memory management system according to an embodiment of the present disclosure. As shown in figure 3, the MMU is between a Packet Memory Unit (PMU), a Traffic Memory Management Unit (TMMU), a CMD_FIFO (Command FIFO, command first in and first out queue), and the HBM (high bandwidth memory)/DDR, the memory management method in the embodiment of the present disclosure mainly achieves the following functions:

1. Being responsible for distributing write requests, write data, and saving of write-release of a multi-queue packet (PK) and a packet descriptor (PD);
2. Being responsible for distributing read requests, and storing read data of the multi-queue PK and the PD;
3. Supporting mode switching of multiple types of memory controllers;
- 4) The mapping of logical addresses to off-chip physical addresses can be reconfigured.

The PMU stores the writing packet, and sends a write release command to a Traffic Memory management unit (TMMU); the TMMU sends a packet descriptor to a queue management unit (QMU); the QMU sends a write command via the TMMU, and transparently transmits write command to the MMU; after storing the write command, the QMU sends a write release signal to the TMMU;

The command queue issues a reading packet command, reads the data of the reading packet and returns the data of the reading packet to the PMU; the TMMU sends a read command and reads the data of the packet descriptor. In the whole framework, in order to solve compatibility and improve off-chip access bandwidth and efficiency, mainly technologies can be adopted are address management, PC equalization, off-chip address mapping and co-packaging technologies, and the compatibility is throughout the whole memory management process, therefore all the functions and performances supported by the support controllers need to be considered.

In an application embodiment, the above memory management method includes:

Step 1, switching of multi controllers. by means of CPU configuration, a memory management module identifies the type of a memory controller externally connected, and an address management sub-module determines an offset address by means of table lookup according to an address area corresponding to the type of the controller, so as to calculate a logical address of the type of controller. For the configuration of different types of memory controllers, and the numbers of gated external connection channels are different. In this disclosure, an off-chip memory interface of 16 channels is designed according to the off-chip memory requirements, and each channel fully supports five channels of write address, write data, write response, read address and read data of an AXI4 bus. In the MMU, there are HBM and DDR modes, and the switching of the supported modes is configurable. the HBM having default 16 Channels can be switched to a DDR mode through a CPU configuration, and it can be connected to different types of DDR4, DDR5, etc. according to the requirements of the user, and as long as the size of the capacity of the connection is not less than a data capacity corresponding to the maximum number of nodes of address management, the connected HBM/DDR space can be effectively utilized;

Since different types of memory controllers are supported, the transmission rate of each type of controller is not necessarily synchronized with the system clock of the MMU, in order to adapt to different controllers and ensure linear speed transmission of data, some memory areas on a chip are divided, in the form of an asynchronous FIFO (first in first out queue), data is stored first, and at the same time, by means of a read-ahead function of self-developed logic, pre-reading data and a command to enter a Ready waiting state, and when a valid memory controller comes, data is sent out at the same period by means of a handshake mechanism, so that the maximum utilization of the bandwidth of an off-chip controller can be ensured, and the data stream is reliably processed across clocks.

Step 2, Configuration of address mapping. An address mapping sub-module reads an address mapping relationship preset by a system according to a configuration type, and the address

mapping relationship can be reconfigured by a CPU according to an application scenario, so as to adjust an optimal mapping manner;

Because the logical address cannot be directly indexed onto the address pin of the HBM/DDR, the logical address is converted into an address that can be accepted by the memory chip after being processed by the MMU, and the accepted address refers to a physical address. Because of the multi-level structure of the memory HBM/DDR, the address mapping mode is closely related to the off-chip read-write bandwidth, the storage rate and the efficiency.

For the address mapping provided by the present disclosure, each physical channel (a data bus bit width of 128 bits) is divided into two virtual channels (pseudo channels, PC for short, and a data bus bit width of 64 bits), and the two virtual channels share a group of addresses and control buses.

Figure 4 is a schematic diagram of an Option E mode of an HBM in a memory management method according to an embodiment of the present disclosure. In the Option E mode of the HBM, as shown in figure 4, each virtual channel corresponds to one controller, and the controller runs at half frequency of a memory. The virtual channel (PS) represented by Psgnt is an internal arbitration signal of the controller, and the controller determines the value of Psgnt according to the physical interface of the PS. When the logic processing is performed, 8 controllers of one HBM stack only need to be treated as 16 controllers, and one control corresponds to one physical PS.

An SID is a unique address of data 8Hi, and can be taken as a bank (storage body) address. The number of banks of 8Hi is twice of that of 4Hi, the 8Hi is 32 banks and the 4Hi is 16 banks. A bank with four continuous IDs belongs to one bank set bank_group. 8Hi and 4Hi are 8 banks and 4 banks, respectively.

The laddr [N:0] is an 8 Byte address (a control bus bit width is 128 bits and divided into two PSs, and the bit width of each PS is 64 bits). Because for the HBM controller, a pre-fetching multiple is 4 and 256 bits are stored at one time, four addresses are occupied, an actual logic of the laddr [1:0] does not allocate an address, and the actual default is 0, and does not used by the controller.

The address sent out by the MMU is 384B (the actual address is sent out at an integral power of 2), the address of the AXI is 1 byte, and each address of the granularity corresponding to the PS channel stores 128 bits data. Therefore, the address correspondence among the above four is as follows:

Figure 5 is a first schematic diagram of address correspondence in a memory management method according to an embodiment of the present disclosure, for Samsung HBM2 4Hi4G, a

storage space of each PS is $4\text{GB}/16 = 2\text{Gb}$, and the address correspondence is as shown in figure 5, $\{A[31:A28], A[4:0]\}$ is filled with 0.

Figure 6 is a second schematic diagram of address correspondence in a memory management method according to an embodiment of the present disclosure, when Samsung HBM2 8Hi8G is used, a storage space of each PS is $8\text{GB}/16=4\text{Gb}$, and the address correspondence is as shown in figure 6, $\{A[31:A29], A[4:0]\}$ is filled with 0.

Figure 7 is a third schematic diagram of address correspondence in a memory management method according to an embodiment of the present disclosure, for Samsung HBM2E 4Hi8G, a storage space of each PS is $8\text{GB}/16 = 4\text{Gb}$, and the address correspondence is shown in figure 7, $\{A[31:A29], A[4:0]\}$ is filled with 0.

Figure 8 is a fourth schematic diagram of address correspondence in a memory management method according to an embodiment of the present disclosure, for Samsung HBM2E 8Hi16G, a storage space of each PS is $8\text{GB}/16=4\text{GB}$, and the address correspondence is shown in figure 8, $\{A[31:A30], A[4:0]\}$ is filled with 0.

Figure 9 is a first schematic diagram of address mapping in a memory management method according to an embodiment of the present disclosure. In order to fully exploit a bandwidth of an HBM, 16 PSs need to be used evenly, and at the same time, bank switching needs to be added in channels. Currently, a Hi4G granularity is used, and a mapping relationship between a logical address and a physical address is shown in figure 9.

Figure 10 is a second schematic diagram of address mapping in a memory management method according to an embodiment of the present disclosure. When a DDR5 is externally mounted, the DDR5 with three channels may be configured and connected, and a mapping relation between a logical address and a physical address of the DDR5 is shown in figure 10.

Figure 11 is a third schematic diagram of address mapping in a memory management method according to an embodiment of the present disclosure, and when a DDR4 is externally mounted, DDR5 with three channels may be configured and connected, and a mapping relationship between a logical address and a physical address of the DDR5 is shown in figure 11.

Step 3, off-chip address management. According to the requirements of the off-chip memory and the structural features of the memory controller, the off-chip memory address is managed in a block unit (configurable fixed-size data unit), which is called virtual address (also called logical address), a data packet sent by the processor to the MMU needs to be segmented in blocks, so as to correspond to a block address, and after such processing, a data packet may correspond to a

plurality of block data, that is, a plurality of block addresses need to be generated. According to different types of the off-chip memory controllers, address ranges for address management are different.

Figure 12 is a schematic diagram of off-chip address management in a memory management method according to an embodiment of the present disclosure. As shown in figure 12, a chunk (relay port) address is 128K, $T[16:13]$ are IDs of 16 large link lists, $T[12:10]$ are IDs of sub-link lists under each of the large link lists, $T[9:0]$ are link list numbers within each of sub-link lists, $B[3:0]$ are the numbers of blocks under each chunk, and $C[2:0]$ are the numbers of slices under each block (data+1 represents the number of slices). The application for the ID of the link list ID adopts RR (Round-Robin) scheduling, and when a flow enters, firstly the RR application selects a large link list, then the RR selects a sub-link list in the large link list, and then applies for a link list from the sub-link list. The same stream will preferentially use up the blocks in one chunk, and different streams will re-apply for a chunk. The $\{T[9:0], B[3:0], T[12:10], T[16:13]\}$ for address management can be taken as a counter, and this addresses change continuously. When an HBM is externally mounted, $T[16:13]$ of 16 channels are in one-to-one correspondence; when a DDR is connected externally, corresponding to three gating channels, $T[14:13]$ are used as channel_IDs, i.e. 0(4, 8, 12), 1(5, 9, 13) and 2(6, 10, 14) channels are gated, and 3, 7, 11 and 15 channels can be configured and unused. Since the granularity of the low two bits of $T[16:13]$ is smaller, so it changes faster, and the memory access efficiency after address mapping will be better. According to a preset bit width size, the MMU can manage $2(4+3+10+4)=2M$ nodes, and can totally manage data of $2M \times 3k=48G$ (the size of blk can be configured, taking 3k as an example).

In order to guarantee the overall performance and bandwidth, it is necessary for the address management module to guarantee the balanced access among the pseudo-channels PCs, avoiding the situation where individual PCs are frequently accessed in a short time and other PCs are idle, and if a certain PC returns slowly, when a feedback is made to the MMU busy state, one scheduling can be reduced according to the real-time command statistics and the single PC historical command statistics, from the whole process analysis of the data stream, the access between PCs is still balanced.

Step 4, The technology of concatenations for small packets. The PD on chip is stored to off ship, the bandwidth and efficiency of the off-chip storage will be reduced theoretically due to the packet length, however, because the storage quantity is relatively large, in order to save on-chip resources, the PD is moved to an off-chip for storage, and storage is performed by using a co-packaging technology, the purpose of co-packaging is to extrude the invalid bytes in the off-chip PK and PD that need to be co-packed through bubble squeezing, and then perform concatenation

on the valid bytes, in this way, the utilization rate of the off-chip memory is improved.

The co-packaging step is that: the MMU receives the off ship PD and PK sent from the PMU, firstly extracts and co-packing PD information, and at the same time squeezes out invalid bytes in the PK, performs shift and concatenations on the small packets, extracts the valid data, then performs co-packing again on the co-packed PD information and the extracted PK of the valid data, and sends the result to the off-chip for storage.

Since the bus bit width of the off-chip is 384B, except for a single packet (the co-packaging is not needed), for different types of co-packaging cases, the results of output at each beat are different, and the co-packaging for small packets includes two cases: small packets co-packaging ($PD_len+PK_len \leq 384B$); small packets co-packaging ($PD_len+PK_len > 384B$). For the first type of co-packaging of small packets, the length of the PD plus the length of the extracted data is less than the bus bit width, thus it can be output at one beat. For the second type of co-packaging of small packets, since the length of the PD plus the length of the extracted data is greater than the bus bit width, it needs to be divided into two beats to output, the output of the first beat is the high 384B, and the remaining part is zero-padded at the tail of the second beat and output, which affects the line speed and needs to be avoided as far as possible.

The solutions provided the present disclosure are not only compatible with various types of memory controllers, but also can improve the off-chip read-write bandwidth and access efficiency. The actual measurement data before and after the improvement of the bandwidth and efficiency in the case of external HBM/DDR5/DDR4 three types of controllers are respectively illustrated through the actual measurement data as below.

The results of testing the externally mounted HBMs using the standard address mapping are shown in table 1. The test results of the efficiency improving method of the present disclosure are as shown in table 2, table 3 is test data of an externally mounted DDR5, table 4 is test data of an externally mounted DDR4, and after comparison and analysis of the actually measured results, the overall bandwidth and storage efficiency of the off-chip in each mode are improved.

Table 1

use case number	length of packet off-chip (Byte)	Total Bandwidth (Gbps)	efficiency
1	128	862	0.35
2	192	1352	0.55

3	256	1643	0.67
4	288	1607	0.65
5	352	1739	0.71
6	384	1769	0.72
7	768	1648	0.67
8	1152	1634	0.67
9	1536	1628	0.66
10	1568	1601	0.65

Table 2

use case number	length of packet off-chip (Byte)	total Bandwidth (Gbps)	efficiency
1	128	1602	0.65
2	192	1899	0.77
3	256	2139	0.87
4	288	2063	0.84
5	352	2186	0.89
6	384	2217	0.90

Table 2 (continued)

use case number	length of packet off-chip (Byte)	Total Bandwidth (Gbps)	efficiency
7	768	2169	0.88
8	1152	2166	0.88

9	1536	2133	0.86
10	1568	2028	0.83

Table 3

use case number	length of packet off-chip (Byte)	Total Bandwidth (Gbps)	efficiency
1	128	136.89	0.59
2	192	154.22	0.67
3	256	163.54	0.71
4	384	168.49	0.73
5	768	170.63	0.74
6	1152	172.45	0.75
7	1184	169.61	0.74
8	1536	173.55	0.75
9	1568	169.95	0.74
10	12288	166.54	0.72

Table 4

use case number	length of packet off-chip (Byte)	Total Bandwidth (Gbps)	efficiency
1	128	200.64	65.35
2	192	247.85	80.73
3	256	267.03	86.98
4	384	272.29	88.69

5	768	270.61	88.15
6	1152	243.78	79.41
7	1536	252.68	82.31
8	1568	232.25	75.65
9	6144	233.99	76.22
10	12288	220.05	71.68

Through the description of the foregoing embodiments, a person skilled in the art may clearly understand that the method according to the foregoing embodiments may be implemented by software in addition to a necessary universal hardware platform, and definitely may also be implemented by hardware. However, in many cases, the former is a preferred implementation. Based on such understanding, the technical solutions of the present disclosure essentially or the part contributing to the prior art may be embodied in the form of a software product, the computer software product is stored in a storage medium (such as a ROM/RAM, a magnetic disk, and an optical disk), and includes several instructions for instructing a terminal device (which may be a mobile phone), a computer, a server, a network device, or the like.

The embodiment further provides a graphics rendering processing device, which is configured to implement the described embodiment and example implementation mode, and what has been described will not be elaborated. The term "module", as used hereinafter, is a combination of software and/or hardware capable of realizing a predetermined function. Although the apparatus described in the following embodiment is preferably implemented by software, implementation of hardware or a combination of software and hardware is also possible and conceived.

Figure 13 is a structural block diagram of a memory management apparatus according to an embodiment of the present disclosure. As shown in figure 13, the apparatus includes:

An identification unit 1302, configured to make an MMU to identify a type of a memory controller externally connected based on CPU configuration information;

A determination unit 1304, configured to, based on the address management sub-module and an address area corresponding to the type of the memory controller, determine an offset address in a manner of table look-up;

A calculation unit 1306, configured to calculate a logic address of the type of the memory

controller based on the offset address, and the numbers of external connection channels gated by different types of memory controllers are different.

By means of the present disclosure, a memory management unit (MMU) is used to identify a memory controller externally connected type based on CPU configuration information; based on the address management sub-module and an address area corresponding to the described memory controller type, determining an offset address in a manner of table look-up; calculating a logic address of the memory controller type based on the offset address; wherein the number of external connection channels gated by different types of memory controllers is different; therefore, switching of different controllers in the same framework is achieved, and the problem of being compatible with different controllers in the same framework can be solved, so that a plurality of controllers can be compatible in the same framework, and the effect of improving the storage efficiency can be achieved.

It should be noted that each module may be implemented by software or hardware. The latter may be implemented in the following manner, but is not limited thereto. All the modules are located in a same processor; alternatively, the modules are located in different processors in an arbitrary combination.

Embodiments of the present disclosure also provide a computer readable storage control program. The computer readable storage control program stores a computer program, wherein the computer program is configured to execute the steps in any one of the described method embodiments when running.

In an exemplary embodiment, the above computer readable storage control program may include, but is not limited to, a CPU and a driver for a storage controller, a control program for connecting an FPGA to an HBM/DDR3/DDR4/DDR5, etc.

An embodiment of the present disclosure also provides a controller, comprising a memory (part of data is stored) and a processor, wherein the controller stores a computer program, and the controller is configured to execute the computer program so as to perform the steps in the above embodiments.

In an exemplary embodiment, the controller can further include a transmission device for protocol conversion, where the transmission device is connected to the controller, and implements a connection to a memory controller.

For specific examples in this embodiment, reference may be made to the examples described in the foregoing embodiments and exemplary embodiments, and details are not repeatedly

described in this embodiment.

Obviously, those skilled in the art should understand that each module or each step of the present disclosure can be implemented by a universal computing device, they may be centralized on a single computing device or distributed on a network composed of a plurality of computing devices, they can be implemented by program codes executable by a computing apparatus, and thus can be stored in a storage apparatus and executed by the computing apparatus, furthermore, in some cases, the shown or described steps may be executed in an order different from that described here, or they are made into integrated circuit modules respectively, or a plurality of modules or steps therein are made into a single integrated circuit module for implementation. As such, the present disclosure is not limited to any particular hardware and software combination.

The foregoing descriptions are merely exemplary embodiments of the present disclosure, but are not intended to limit the present disclosure. For those skilled in the art, the present disclosure may have various modifications and variations. Any modifications, equivalent replacements, improvements and the like made within the principle of the present disclosure shall belong to the scope of protection of the present disclosure.

Claims:

1. A memory management method, comprising:

identifying, by a Memory Management Unit (MMU), a type of a memory controller externally connected based on configuration information of a Central Processing Unit (CPU);

based on an address management sub-module and an address area corresponding to the type of the memory controller, determining an offset address in a manner of table look-up;

calculating a logical address of the type of the memory controller based on the offset address;
wherein the numbers of external connection channels gated by different types of memory controllers are different.
2. The method according to claim 1, wherein the method further comprises:

reading, by an address mapping submodule of the MMU, a preset address mapping relationship, and the address mapping relationship is for converting an on-chip logical address into a physical address that can be accepted by a memory chip.
3. The method according to claim 1, wherein the method further comprises:

reading, by an address mapping submodule of the MMU, a preset address mapping relationship, and the address mapping relationship is for converting an on-chip logic address into a physical address that can be accepted by a memory chip;

performing, according to different application scenarios, a reconfiguration by means of the CPU, so as to acquire a reconfigured address mapping relationship corresponding to the application scenario.
4. The method according to claim 1, wherein the method further comprises:

according to an off-chip memory requirement and a structural attribute of the memory controller, a data packet sent by the CPU to the MMU is segmented according to a block unit, wherein the segmented data packets corresponds to block addresses, one data packet corresponds to a plurality of block addresses, and managed address ranges are different when the types of the memory controller are different.
5. The method according to claim 1, wherein the method further comprises:

receiving, by the MMU, off-chip data sent by a packet memory unit (PMU), wherein the off-chip data comprises a packet descriptor (PD) and a packet (PK);

extracting data information of the PD;

co-packing the data information to obtain co-packed data, and deleting a first data in the PK;

performing shift and concatenations on the co-packed data according to effective data bits, and extracting second data; and

co-packing the data information of the PD in the co-packed data and the extracted second data to obtain target co-packed data

sending the target co-packed data to an off-chip memory for storage.

6. The method according to claim 5, wherein the method further comprises:

outputting the PD and the second data at a same time when a data bit length of the PD plus a data bit length of the extracted second data is less than or equal to a bus bit width;

splitting the PD and the second data into first split data and second split data when the data bit length of the PD plus the extracted data bit length of the second data is greater than the bus bit width;

outputting the PD and the first split data;

performing a zero-padding operation on the second split data to obtain zero-padding processed data with a data bit length equal to the bus bit width, and outputting the zero-padding processed data.

7. The method according to any one of claims 1 to 6, wherein the method further comprises:

issuing, by a packet memory unit (PMU), a writing packet;

storing, by the PMU, the writing packet, and sending a write release command to a Traffic Memory management unit (TMMU), and sending, by the TMMU, a Packet Descriptor to a queue management unit (QMU);

issuing, by the QMU, a write command via the TMMU, and transparently transmits the write command to the MMU, and after storing the write command, sending, by the QMU, a write release signal to the TMMU;

issuing, by a command queue, a reading packet command, reading the data corresponding to the reading packet command and returning the data corresponding to the reading packet command to the PMU;

sending, by the TMMU, a read command, and reading the data of the packet descriptor.

8. A memory management apparatus, comprising:

an identification unit, configured to enable a memory management unit (MMU) to identify, based on configuration information of a central processing unit (CPU), a type of a memory controller externally connected;

a determination unit, configured to determine an offset address in a manner of table look-up based on an address management sub-module and an address area corresponding to the type of the memory controller;

a calculation unit, configured to calculate a logic address of the type of the memory controller based on the offset address, wherein the numbers of external connection channels gated by different types of memory controllers are different.

9. A computer-readable storage control program, wherein the computer-readable storage control program stores a computer program, wherein the computer program is configured to perform, when executed by a processor, the method as claimed in any one of claims 1 to 7.

10. A controller, comprising a memory and a processor, wherein a computer program is stored in the memory, and the processor is configured to execute the computer program to perform the method as claimed in any one of claims 1 to 7.