



- (51) **International Patent Classification:**
G06F 13/16 (2006.01) *G06F 13/42* (2006.01)
- (21) **International Application Number:**
PCT/IB2013/000521
- (22) **International Filing Date:**
25 January 2013 (25.01.2013)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
61/591,705 27 January 2012 (27.01.2012) US
- (71) **Applicant:** MARVELL WORLD TRADE LTD.
[BB/BB]; L'Horizon, Gunsite Road, Brittons Hill, St. Michael, BB14027 (BB).
- (72) **Inventors:** ZHU, Jun; 1254 Regency Drive, San Jose, CA 95129 (US). CAO, Joseph, Jun; 15 Grove Street, Los Gatos, CA 95030 (US). YANG, Tsung-Ju; 2845 Rubino Circle, San Jose, CA 95125 (US). LU, Ruoyang; 2931 Queens Estates Ct., San Jose, CA 95135 (US).
- (74) **Agents:** SAUER, Joseph, M. et al.; Jones Day, 901 Lakeside Avenue, Cleveland, OH 44114 (US).

(81) **Designated States** (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, 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, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

Published:

— without international search report and to be republished upon receipt of that report (Rule 48.2(g))

(54) **Title:** SYSTEMS AND METHODS FOR DYNAMIC PRIORITY CONTROL

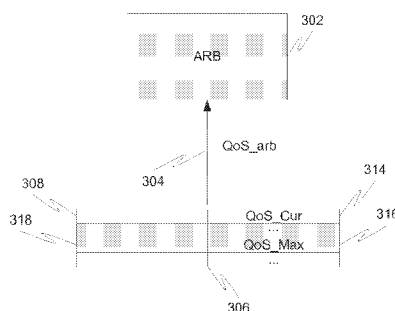


FIG. 3

(57) **Abstract:** System and methods are provided for dynamically managing a first-in/first-out (FIFO) command queue of a system controller. One or more commands are received into the command queue, a command being associated with a priority parameter. A current command first in line to be executed in the command queue is determined, the current command being associated with a first priority parameter. A second command associated with a second priority parameter is determined, the second priority parameter being largest among priority parameters associated with the one or more commands. A final priority parameter for the current command is computed based at least in part on the second priority parameter.



SYSTEMS AND METHODS FOR DYNAMIC PRIORITY CONTROL

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This disclosure claims priority to and benefit from U.S. Provisional Patent Application No. 61/591,705, filed on January 27, 2012, the entirety of which is incorporated herein by reference.

FIELD

[0002] The technology described in this patent document relates generally to data processing and more particularly to priority control in data processing.

BACKGROUND

[0003] A memory system often includes semiconductor memory devices, such as dynamic random access memory (DRAM), synchronous DRAM (SDRAM), double data rate (DDR, DDR2, DDR3, etc.) SDRAM, etc. Various source devices, such as processors, peripheral devices (e.g., input/output devices), audio and video devices, may generate memory operation commands, including read memory operations to transfer data from memory devices to the source devices and write memory operations to transfer data from the source devices to the memory devices. Usually, a memory controller is implemented to receive the memory operation commands from the source devices and to control the memory devices to perform memory operations in response to the commands. The memory controller often includes command queues to capture the memory operation commands.

[0004] Priority parameters (e.g., Quality of Service (QoS) parameters) of the memory operation commands may be transmitted as parts of the commands to the memory controller.

The memory controller may arbitrate among memory operation commands from different command queues and schedule execution of such commands based on their respective priority parameters. FIG. 1 illustrates an example of a memory controller scheduling execution of memory operation commands. An arbiter component 108 in a memory controller 100 schedules execution of memory operation commands 104 from multiple command queues 102 based on priority parameters 106 of the memory operation commands 104. As shown in FIG. 1, the memory controller 100 includes multiple system interface ports (SIPs) 110 which correspond to multiple command queues 102 respectively. A command queue stores one or more memory operation commands 104 which each include a priority parameter 106 (e.g., QoS). Each command queue has a current command which is at the top of the command queue and thus first in line to be serviced. The arbiter component 108 compares the priority parameters (e.g., QoS) of the current commands in different command queues, and selects one current command with a highest priority parameter to be serviced. For example, a command queue often operates in a first-in-first-out (FIFO) manner. That is, a current command of a command queue is the one that is received earlier than other commands in the command queue.

SUMMARY

[0005] In accordance with the teachings described herein, systems and methods are provided for dynamically managing a first-in/first-out (FIFO) command queue of a system controller. One or more commands are received into the command queue, a command being associated with a priority parameter. A current command first in line to be executed in the command queue is determined, the current command being associated with a first priority parameter. A second command associated with a second priority parameter is determined, the second priority parameter being largest among priority parameters associated with the one or more commands.

A final priority parameter for the current command is computed based at least in part on the second priority parameter.

[0006] In another embodiment, an integrated circuit for dynamically managing a first-in/first-out (FIFO) command queue of a system controller includes, an interface circuit configured to receive one or more commands into the command queue, a command being associated with a priority parameter, a monitoring circuit configured to determine a current command first in line to be executed in the command queue, the current command being associated with a first priority parameter, and determine a second command associated with a second priority parameter, the second priority parameter being largest among priority parameters associated with the one or more commands, and a selection circuit configured to compute a final priority parameter for the current command based at least in part on the second priority parameter and output the final priority parameter in order for the current command to be selected for execution when the final priority parameter satisfies a predetermined condition.

[0007] In yet another embodiment, a system for dynamically managing a first-in/first-out (FIFO) command queue of a system controller includes one or more data processors, and a computer-readable memory encoded with programming instructions for commanding the one or more data processors to perform steps. The steps include, receiving one or more commands into the command queue, a command being associated with a priority parameter, determining a current command first in line to be executed in the command queue, the current command being associated with a first priority parameter, and determining a second command associated with a second priority parameter, the second priority parameter being largest among priority parameters associated with the one or more commands. The steps further include computing a final priority parameter for the current command based at least in part on the second priority parameter, and

outputting the final priority parameter in order for the current command to be selected for execution when the final priority parameter satisfies a predetermined condition.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] FIG. 1 illustrates an example of a memory controller scheduling execution of memory operation commands.

[0009] FIG. 2 illustrates an example of a FIFO command queue.

[0010] FIG. 3 illustrates an example of generating dynamic priority parameters for commands in a command queue.

[0011] FIG. 4 illustrates another example of generating dynamic priority parameters for commands in a command queue.

[0012] FIG. 5 illustrates example data fields of commands in a command queue for generating dynamic priority parameters.

[0013] FIG. 6 illustrates an example of a memory controller scheduling execution of memory operation commands based on dynamic priority parameters associated with command queues.

DETAILED DESCRIPTION

[0014] Referring back to FIG. 1, the arbiter component 108 selects one of multiple current commands which has a highest priority parameter to be serviced. Thus, if a current command of a particular command queue has a low priority parameter, then such a current command may need to wait for a long period of time before it can be serviced. Other commands in the command queue are blocked by the current command, even though they may have high priority parameters.

[0015] FIG. 2 illustrates an example of a FIFO command queue. Commands with high priority parameters (e.g., command 204) are blocked by a current command 202 with a low priority parameter. As shown in FIG. 2, a memory operation command includes an identification number ("ID") for ordering control, an address ("Addr") indicating a memory location for accessing data in the memory, and a priority parameter ("QoS") indicating how urgent the command is. A memory operation command 202 with a low priority parameter "1" (e.g., QoS) stays at the top of the command queue 200 and is the current command of the command queue 200. Because the current command 202 has a low priority parameter, it may not be serviced for a long time. Thus, even though other commands in the command queue 200 may have high priority parameters, they cannot get serviced. For example, another memory operation command 204 has a very high priority parameter "15" (e.g., QoS). However, the command 204 is in the middle of the command queue 200, and thus it will not have a chance to be serviced until all commands before the command 204 have been serviced.

[0016] As an example, a Liquid Crystal Display (LCD) controller sends commands to read data from a memory. At first, a LCD buffer has enough data to be displayed, and the LCD controller sends read commands with low priority parameters (e.g., QoS) to a command queue associated with the LCD. The memory controller does not service these read commands in time because commands from other command queues may have higher priority parameters. Later when the buffer does not have enough data to be displayed, the LCD controller sends read commands with high priority parameters to the same command queue associated with the LCD. The previous read commands with low priority parameters are still in the command queue waiting for execution, and block the subsequent read commands with high priority parameters. Then, error may occur when the buffer has no data to be displayed.

[0017] A virtual channel approach or a multi-channel approach which often uses multiple physical command queues for a particular system interface port may ameliorate the problem, since commands with different priority parameters may be input into different command queues and the commands with high priority parameters may not be blocked by the commands with low priority parameters. However, the implementation of the virtual channel approach or the multi-channel approach is very expensive. In addition, such virtual channel approach or multi-channel approach will typically encounter a different problem.

[0018] Often, a source device needs to access a number of consecutive locations of the memory. For each location, the source device usually sends out a command. These commands from the source device share a same identification number. Usually, it is preferred to execute these commands in the order that they are sent out, so that the target locations of the memory can be accessed consecutively. A single FIFO command queue for a particular system interface port can often achieve this without any problem because the commands received first will be serviced first. However, under the virtual channel approach or the multi-channel approach, commands with the same identification number are often sent to different physical command queues. Additional mechanisms are usually needed to execute commands with the same identification number in order, which will increase the complexity and cost of the system.

[0019] The present disclosure presents an approach allowing commands in a command queue to be serviced in time according to the status of the command queue. FIG. 3 illustrates an example of generating dynamic priority parameters for commands in a command queue. An arbiter component 302 receives a dynamic priority parameter 304 ("QoS_arb") determined based on the status of a command queue 306. If the dynamic priority parameter 304 is higher than other priority parameters associated with other command queues, the arbiter component 302

selects a current command 308 of the command queue 306 to be serviced. When commands with high priority parameters are received into the command queue 306 later than the current command 308, the dynamic priority parameter 304 is increased to speed up the service of the command queue 306. When the commands with high priority parameters are serviced, the dynamic priority parameter 304 is reduced to slow down the service of the command queue 306.

[0020] Specifically, an algorithm may be implemented to dynamically determine a highest priority parameter in the command queue 306. How long the command with the highest priority parameter has stayed in the command queue 306 may be taken into account to determine the dynamic priority parameter 304. As an example, a command 318 is determined to have a highest priority parameter 316 ("QoS_Max") in the command queue 306. If the command 318 has stayed in the command queue 306 longer than a wait-time threshold, the dynamic priority parameter 304 is determined to be equal to the highest priority parameter 316 ("QoS_Max"). On the other hand, if the command 318 has stayed in the command queue 306 no longer than the wait-time threshold, the dynamic priority parameter 304 is determined to be equal to half of a sum of the highest priority parameter 316 ("QoS_Max") and a current priority parameter 314 of a current command 308. Alternatively, in some circumstances, the dynamic priority parameter 304 is determined to be equal to the highest priority parameter 316 ("QoS_Max") regardless of how long the command 318 has stayed in the command queue 306.

[0021] FIG. 4 illustrates another example of generating dynamic priority parameters for commands in a command queue. As shown in FIG. 4, a selection component 610 (e.g., a programmable register) outputs a signal 622 ("QoS_sel") to a multiplexer 612 to select one of three modes for generating a dynamic priority parameter 604 for a command queue 606. Under a first mode, the dynamic priority parameter 604 is always determined to be equal to a current

priority parameter 614 of a current command 608 in the command queue 606. Under a second mode, the dynamic priority parameter 604 is always determined to be equal to a highest priority parameter 616 in the command queue 606. Further, under a third mode, the multiplexer 612 outputs a modified priority parameter 620 ("QoS") as the dynamic priority parameter 604.

[0022] For example, the modified priority parameter 620 may be determined based on how long a command 618 with the highest priority parameter 616 has stayed in the command queue 606. If the command 618 has stayed in the command queue 606 longer than a first wait-time threshold, the modified priority parameter 620 is determined to be equal to the maximum priority parameter 616. On the other hand, if the command 618 has stayed in the command queue 606 no longer than the first wait-time threshold, the modified priority parameter 620 is determined to be equal to half of a sum of the maximum priority parameter 616 and the current priority parameter 614.

[0023] Further, how long the current command 608 has stayed in the command queue 606 may also be taken into account to determine the modified priority parameter 620. As an example, if the command 618 has stayed in the command queue 606 longer than the first wait-time threshold and the current command 608 has stayed in the command queue 606 longer than a second wait-time threshold, the modified priority parameter 620 is determined to be equal to a first value. If the command 618 has stayed in the command queue 606 no longer than the first wait-time threshold and the current command 608 has stayed in the command queue 606 longer than a second wait-time threshold, the modified priority parameter 620 is determined to be equal to a second value. If the command 618 has stayed in the command queue 606 longer than the first wait-time threshold and the current command 608 has stayed in the command queue 606 no longer than a second wait-time threshold, the modified priority parameter 620 is determined to

be equal to a third value. In addition, if the command 618 has stayed in the command queue 606 no longer than the first wait-time threshold and the current command 608 has stayed in the command queue 606 no longer than a second wait-time threshold, the modified priority parameter 620 is determined to be equal to a fourth value. For example, the first value and the third value are equal to the maximum priority parameter 616, and the second value and the fourth value are equal to half of the sum of the maximum priority parameter 616 and the current priority parameter 614.

[0024] FIG. 5 illustrates example data fields of commands in a command queue for generating dynamic priority parameters. Each command in a command queue 400 includes three data fields related to generating dynamic priority parameters – a validity factor (“V”) indicating whether the command is valid, a wait-time factor (“WT”) indicating a wait time of the command (i.e., how long the command stays in the command queue 400), and an original priority parameter (“QoS_org”). For example, when the validity factor of a command is 1, the command is valid, and when the validity factor is 0, the command is invalid. The wait-time factor of a valid command begins to increase when the command is received into the command queue 400 until reaching a maximum value, and is cleared when the command is popped out of the command queue 400. A read pointer 410 (“rd_ptr”) points to a current command 412, and increases by one when the current command 412 is popped out of the command queue 400. A write pointer 408 (“wr_ptr”) points to a next available location in the command queue 400 for receiving a new command, and increases by one when a new command is received. As an example, the command queue 400 is managed in a circular FIFO manner.

[0025] A two-dimensional array, QoS_Info[Q_Size-1:0][Entry_Size-1:0], may be defined to store information of the above-noted data fields for generating dynamic priority parameters,

where Q_Size indicates how many commands can be stored in the command queue 400, and Entry_Size represents a sum of sizes of a validity factor, a wait-time factor and an original priority parameter.

[0026] A maximum priority parameter of valid commands in the command queue 400 can be determined as follows:

```

QoS_max=0; max_loc=0;
For (i=0; i<Q_Size; i++){
    if ((QoS_max<QoS_Info[i].QoS_org) & (QoS_info[i].V==1))
        QoS_max=QoS_info[i].QoS_org, max_loc=i;
}

```

[0027] A wait-time factor of a command having the maximum priority parameter is determined as follows:

$$WT\_Max\_QoS = QoS\_Info[max\_loc].WT$$

A wait-time factor of the current command is determined as follows:

$$WT\_Cur = QoS\_Info[rd\_ptr].WT$$

[0028] For the first mode as discussed in FIG. 3, the dynamic priority parameter is determined as follows:

$$QoS' = QoS\_Info[rd\_ptr].QoS\_org$$

For the second mode, the dynamic priority parameter is determined as follows:

$$QoS' = QoS\_max$$

In addition, for the third mode, the dynamic priority parameter is determined based on the first wait-time threshold ("THR1") and the second wait-time threshold ("THR2") as follows:

WT_Max_QoS>THR1	WT_Cur>THR2	QoS'
-----------------	-------------	------

Yes	Yes	QoS_Max
No	Yes	$(QoS\_Max + QoS\_Cur)/2$
Yes	No	QoS_Max
No	No	$(QoS\_Max + QoS\_Cur)/2$

[0029] FIG. 6 illustrates an example of a memory controller scheduling execution of memory operation commands based on dynamic priority parameters associated with command queues. An arbiter component 502 in a memory controller 500 schedules execution of memory operation commands from multiple command queues 504 based on dynamic priority parameters 506 (“QoS_arb”) associated with the command queues 504 respectively. The arbiter component 502 compares the dynamic priority parameters 506 (“QoS_arb”) associated with the command queues 504, and selects, through a multiplexer 510, a current command of a command queue that has a highest dynamic priority parameter. The selected current command is output to a memory command scheduler 512 (e.g., a DDR command scheduler) to be serviced. The command queues 504 correspond to multiple system interface ports (SIPs) 508 respectively.

[0030] This written description uses examples to disclose the invention, include the best mode, and also to enable a person skilled in the art to make and use the invention. The patentable scope of the invention may include other examples that occur to those skilled in the art. For example, the systems and methods described herein may be implemented for priority control in any system controller with a single-command-queue structure. As an example, the systems and methods described herein may be implemented for priority control in modules or components of a system-on-a-chip (SOC), such as SOC fabrics (bus interconnects), PCIe modules, and USB modules in the SOC.

[0031] For example, the systems and methods described herein may be implemented on many different types of processing devices by program code comprising program instructions that are executable by the device processing subsystem. Other implementations may also be used, however, such as firmware or appropriately designed hardware configured to carry out the methods and systems described herein. In another example, the systems and methods described herein may be implemented in an independent processing engine, as a co-processor, or as a hardware accelerator. In yet another example, the systems and methods described herein may be provided on many different types of computer-readable media including computer storage mechanisms (e.g., CD-ROM, diskette, RAM, flash memory, computer's hard drive, etc.) that contain instructions (e.g., software) for use in execution by a processor to perform the methods' operations and implement the systems described herein.

CLAIMS

What is claimed is:

1. A method for dynamically managing a first-in/first-out (FIFO) command queue of a system controller, the method comprising:

receiving one or more commands into the command queue, a command being associated with a priority parameter;

determining a current command first in line to be executed in the command queue, the current command being associated with a first priority parameter;

determining a second command associated with a second priority parameter, the second priority parameter being largest among priority parameters associated with the one or more commands;

computing a final priority parameter for the current command based at least in part on the second priority parameter; and

outputting the final priority parameter in order for the current command to be selected for execution when the final priority parameter satisfies a predetermined condition.

2. The method of claim 1, and further comprising:

computing a second final priority parameter for a second current command in a second FIFO command queue in the system controller;

wherein if the final priority parameter is larger than the second final priority parameter, the final priority parameter satisfies the predetermined condition so that the current command is selected for execution.

3. The method of claim 1, and further comprising:

computing a second final priority parameter for a second current command in a second FIFO command queue in the system controller;

computing a third final priority parameter for a third current command in a third FIFO command queue in the system controller;

wherein if the final priority parameter is larger than both the second final priority parameter and the third final priority parameter, then the final priority parameter satisfies the predetermined condition so that the current command is selected for execution.

4. The method of claim 1, wherein:

the current command is associated with a first wait-time parameter, the first wait-time parameter indicating a duration of the current command in the command queue; and

the second command is associated with a second wait-time parameter, the second wait-time parameter indicating a duration of the second command in the command queue.

5. The method of claim 4, wherein:

when the second wait-time parameter is larger than a predetermined threshold, the final priority parameter is computed to be equal to a first value; and

when the second wait-time parameter is smaller than or equal to the predetermined threshold, the final priority parameter is computed to be equal to a second value.

6. The method of claim 5, wherein the first value is equal to the second priority parameter, and the second value is equal to half of a sum of the second priority parameter and the first priority parameter.

7. The method of claim 4, wherein:
when the second wait-time parameter is larger than a first threshold and the first wait-time parameter is larger than a second threshold, the final priority parameter is computed to be equal to a first value;

when the second wait-time parameter is smaller than or equal to the first threshold and the first wait-time parameter is larger than the second threshold, the final priority parameter is computed to be equal to a second value;

when the second wait-time parameter is larger than the first threshold and the first wait-time parameter is smaller than or equal to the second threshold, the final priority parameter is computed to be equal to a third value; and

when the second wait-time parameter is smaller than or equal to the first threshold and the first wait-time parameter is smaller than or equal to the second threshold, the final priority parameter is computed to be equal to a fourth value.

8. The method of claim 1, wherein the final priority parameter is computed to be equal to the second priority parameter.

9. An integrated circuit for dynamically managing a first-in/first-out (FIFO) command queue of a system controller, the integrated circuit comprising:

an interface circuit configured to receive one or more commands into the command queue, a command being associated with a priority parameter;

a monitoring circuit configured to determine a current command first in line to be executed in the command queue, the current command being associated with a first priority parameter, and determine a second command associated with a second priority parameter, the second priority parameter being largest among priority parameters associated with the one or more commands; and

a selection circuit configured to compute a final priority parameter for the current command based at least in part on the second priority parameter and output the final priority parameter in order for the current command to be selected for execution when the final priority parameter satisfies a predetermined condition.

10. The integrated circuit of claim 9, and further comprising:

a second selection circuit configured to compute a second final priority parameter for a second current command in a second FIFO command queue in the system controller;

wherein if the final priority parameter is larger than the second final priority parameter, the final priority parameter satisfies the predetermined condition so that the current command is selected for execution.

11. The integrated circuit of claim 9, and further comprising:

a second selection circuit configured to compute a second final priority parameter for a second current command in a second FIFO command queue in the system controller;

a third selection circuit configured to compute a third final priority parameter for a third current command in a third FIFO command queue in the system controller;

wherein if the final priority parameter is larger than both the second final priority parameter and the third final priority parameter, the final priority parameter satisfies the predetermined condition so that the current command is selected for execution.

12. The integrated circuit of claim 9, wherein:

the current command is associated with a first wait-time parameter, the first wait-time parameter indicating a duration of the current command in the command queue; and

the second command is associated with a second wait-time parameter, the second wait-time parameter indicating a duration of the second command in the command queue.

13. The integrated circuit of claim 12, wherein:

when the second wait-time parameter is larger than a predetermined threshold, the selection circuit is further configured to compute the final priority parameter to be equal to a first value; and

when the second wait-time parameter is smaller than or equal to the predetermined threshold, the selection circuit is further configured to compute the final priority parameter to be equal to a second value.

14. The integrated circuit of claim 12, wherein:

when the second wait-time parameter is larger than a first threshold and the first wait-time parameter is larger than a second threshold, the selection circuit is further configured to compute the final priority parameter to be equal to a first value;

when the second wait-time parameter is smaller than or equal to the first threshold and the first wait-time parameter is larger than the second threshold, the selection circuit is further configured to compute the final priority parameter to be equal to a second value;

when the second wait-time parameter is larger than the first threshold and the first wait-time parameter is smaller than or equal to the second threshold, the selection circuit is further configured to compute the final priority parameter to be equal to a third value; and

when the second wait-time parameter is smaller than or equal to the first threshold and the first wait-time parameter is smaller than or equal to the second threshold, the selection circuit is further configured to compute the final priority parameter to be equal to a fourth value.

15. The integrated circuit of claim 9, wherein the selection circuit is further configured to compute the final priority parameter to be equal to the second priority parameter.

16. A system for dynamically managing a first-in/first-out (FIFO) command queue of a system controller, the system comprising:

one or more data processors;

a computer-readable memory encoded with programming instructions for commanding the one or more data processors to perform steps comprising:

receiving one or more commands into the command queue, a command being associated with a priority parameter;

determining a current command first in line to be executed in the command queue, the current command being associated with a first priority parameter;

determining a second command associated with a second priority parameter, the second priority parameter being largest among priority parameters associated with the one or more commands;

computing a final priority parameter for the current command based at least in part on the second priority parameter; and

outputting the final priority parameter in order for the current command to be selected for execution when the final priority parameter satisfies a predetermined condition.

17. The system of claim 16, wherein the programming instructions encoded in the computer-readable memory are adapted for commanding the one or more data processors to perform further steps comprising:

computing a second final priority parameter for a second current command in a second FIFO command queue in the system controller;

wherein if the final priority parameter is larger than the second final priority parameter, the final priority parameter satisfies the predetermined condition so that the current command is selected for execution.

18. The system of claim 16, wherein the programming instructions encoded in the computer-readable memory are adapted for commanding the one or more data processors to perform further steps comprising:

computing a second final priority parameter for a second current command in a second FIFO command queue in the system controller;

computing a third final priority parameter for a third current command in a third FIFO command queue in the system controller;

wherein if the final priority parameter is larger than both the second final priority parameter and the third final priority parameter, the final priority parameter satisfies the predetermined condition so that the current command is selected for execution.

19. The system of claim 16, wherein:

the current command is associated with a first wait-time parameter, the first wait-time parameter indicating a duration of the current command in the command queue; and

the second command is associated with a second wait-time parameter, the second wait-time parameter indicating a duration of the second command in the command queue.

20. The system of claim 19, wherein:

when the second wait-time parameter is larger than a predetermined threshold, the final priority parameter is computed to be equal to a first value; and

when the second wait-time parameter is smaller than or equal to the predetermined threshold, the final priority parameter is computed to be equal to a second value.

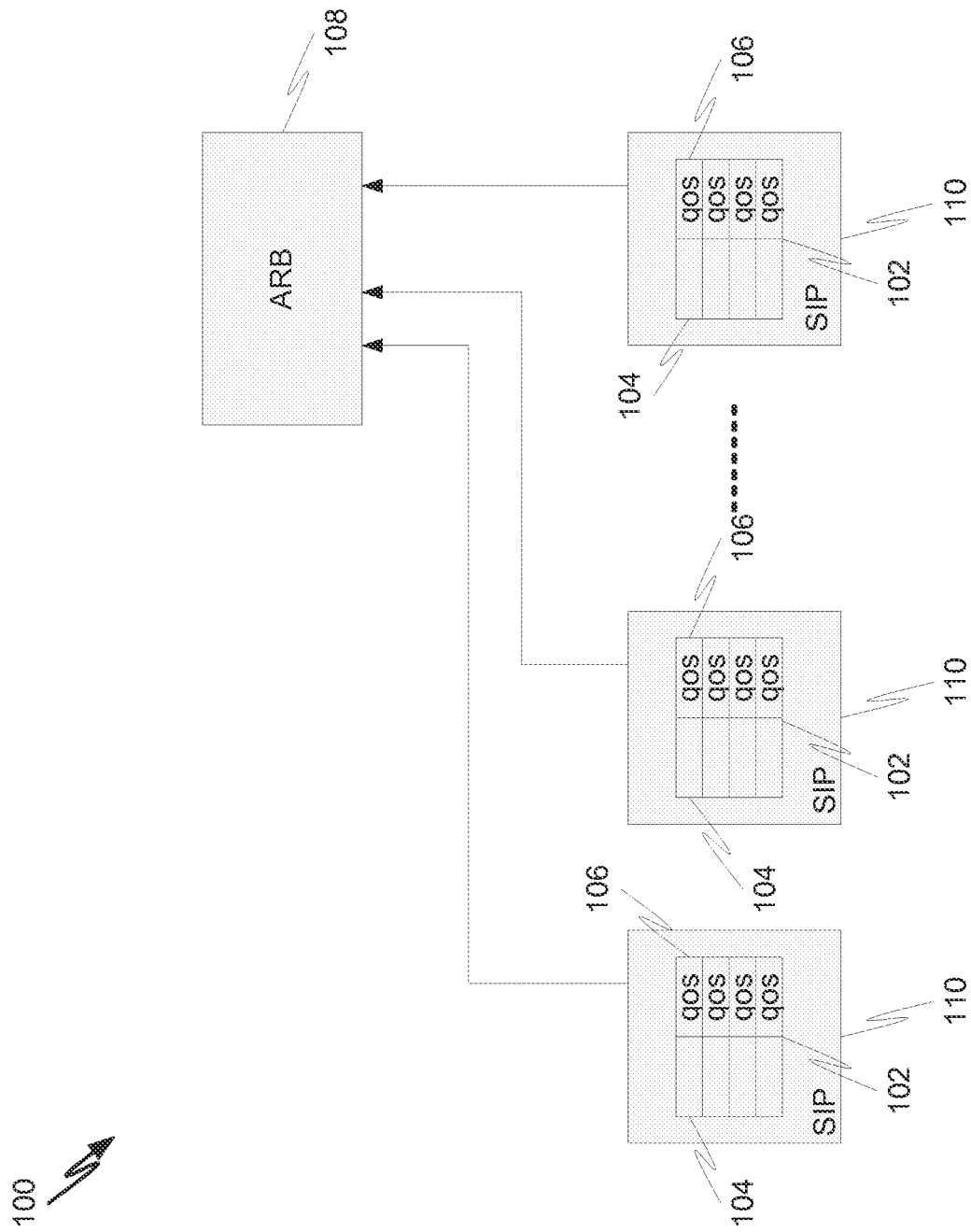


FIG. 1 (Prior Art)

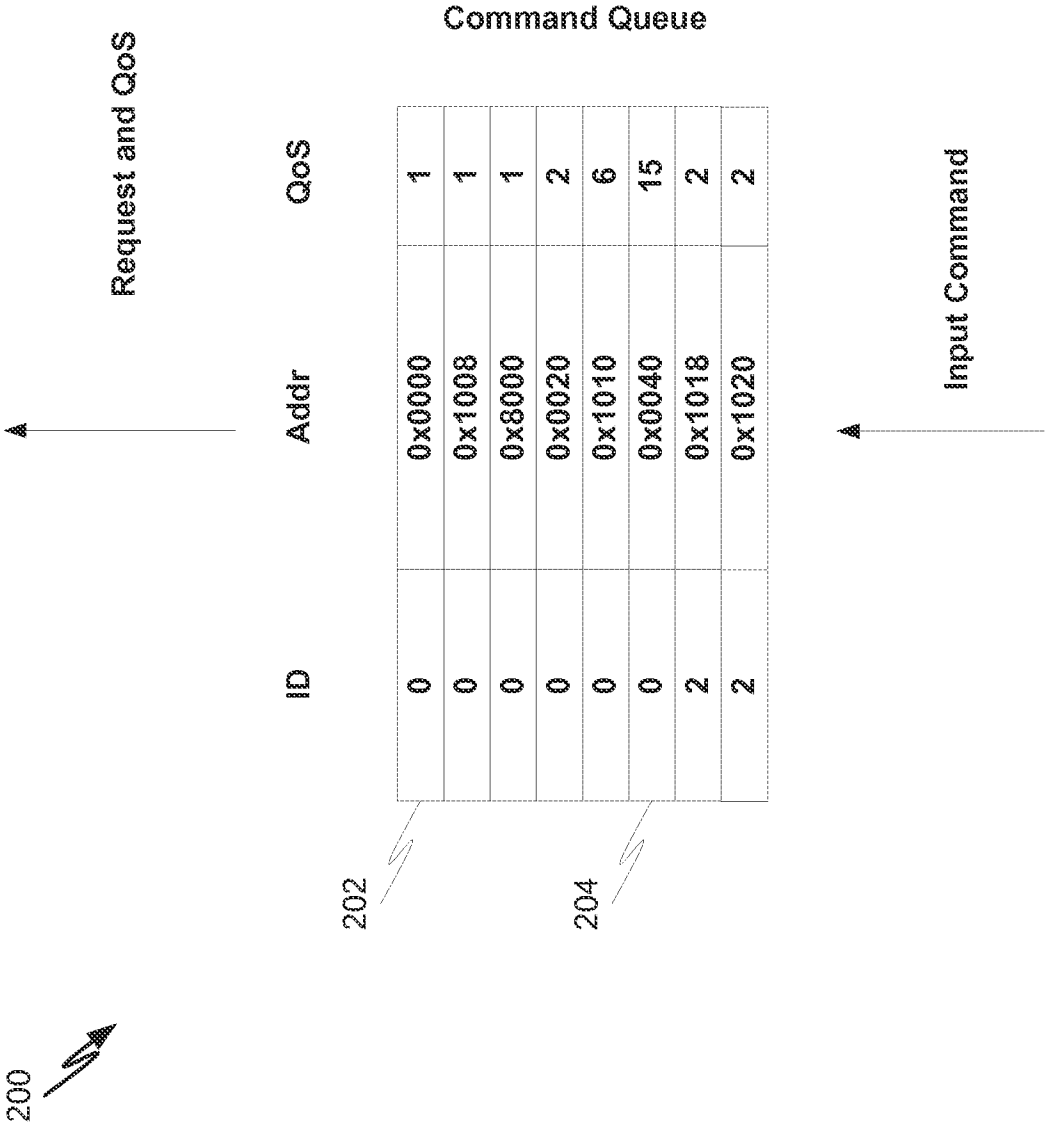


FIG. 2 (Prior Art)

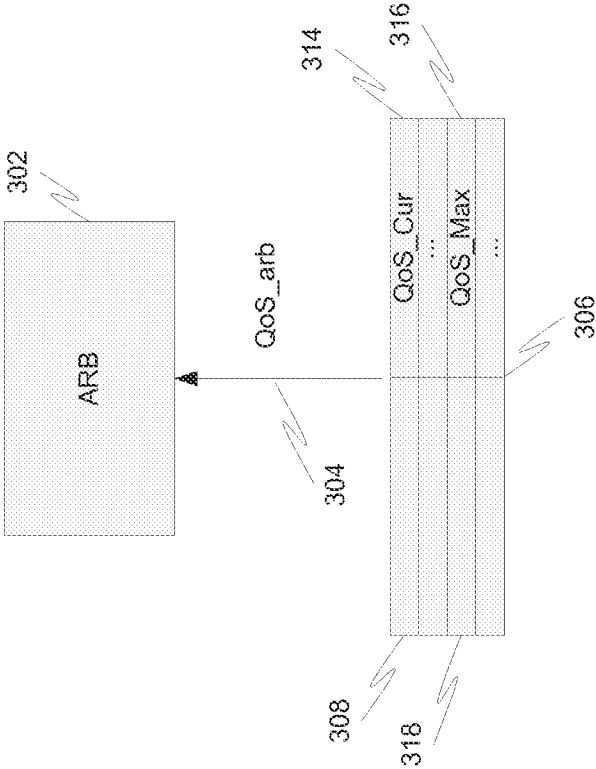
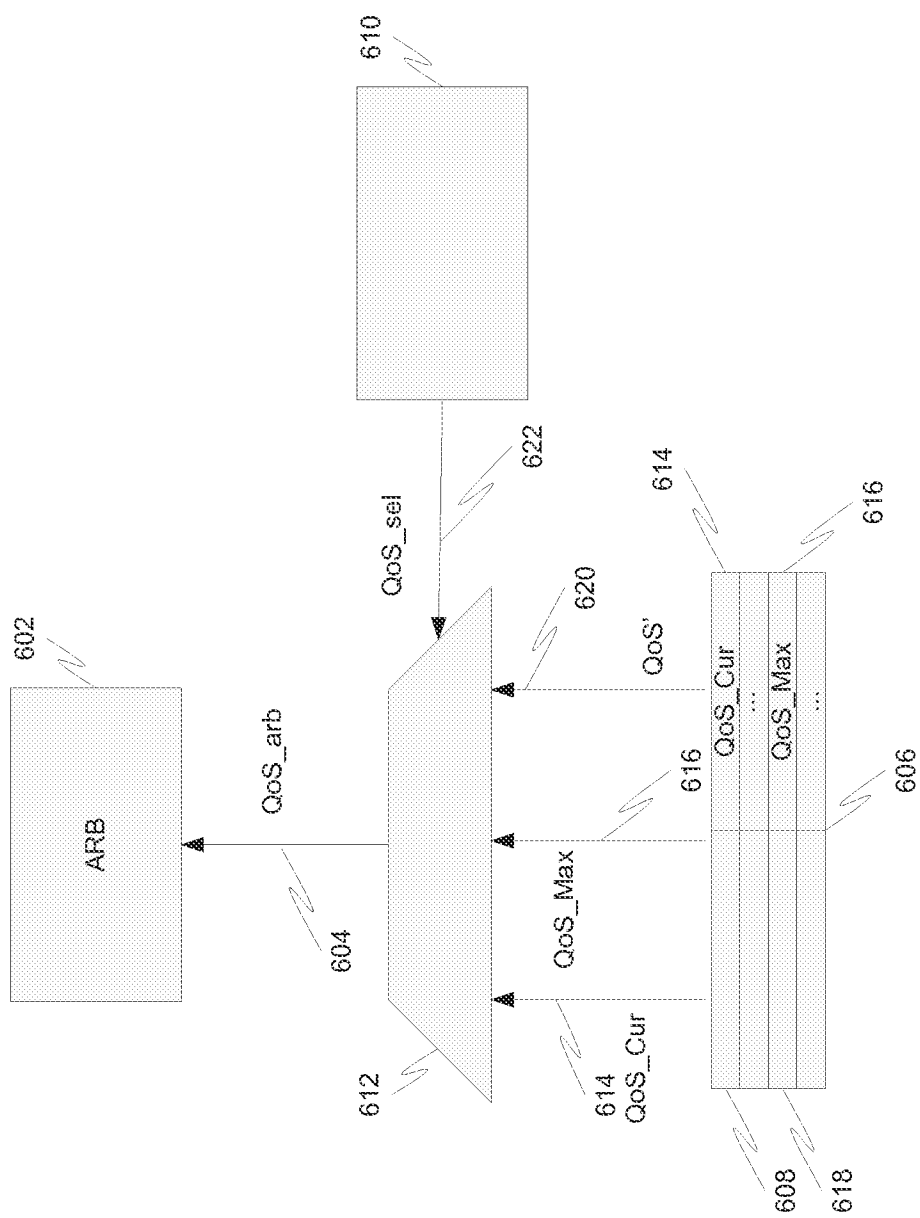


FIG. 3



১৫৫

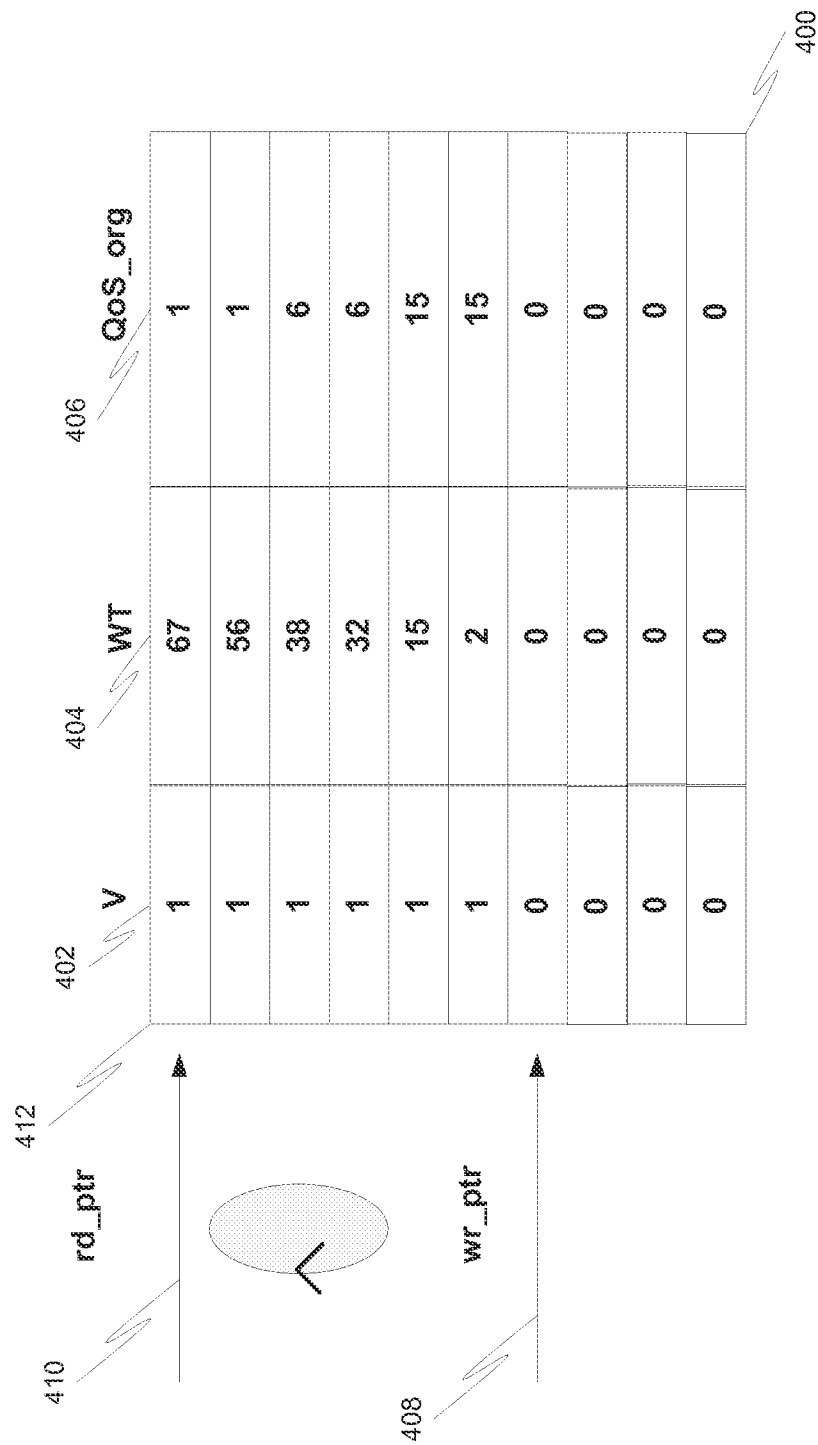


FIG. 5

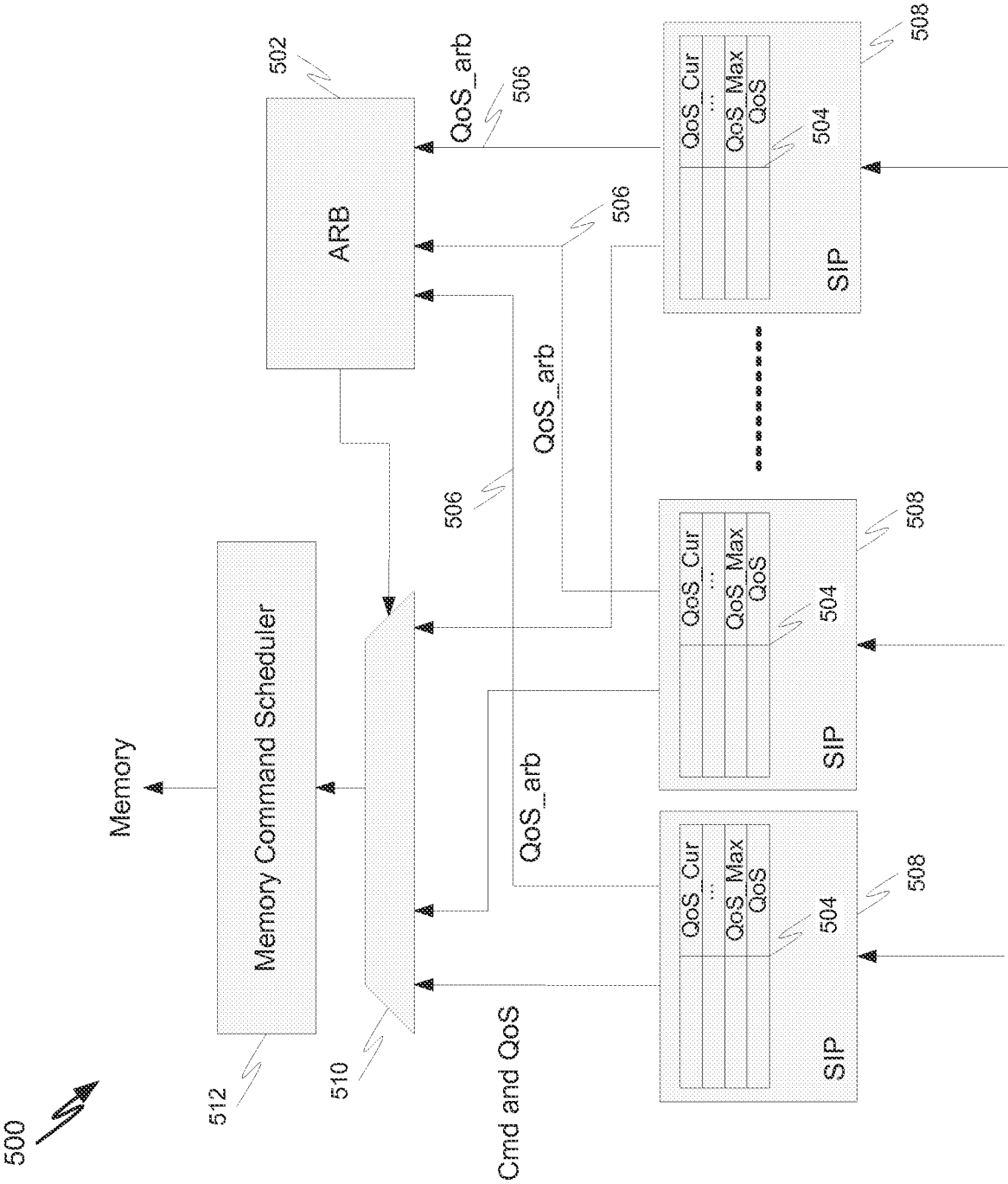


FIG. 6