US 20140019730A1

(19) **United States**

(12) **Patent Application Publication**

Li et al.

(10) **Pub. No.: US 2014/0019730 A1**

(43) **Pub. Date: Jan. 16, 2014**

(54) **METHOD AND DEVICE FOR DATA TRANSMISSION BETWEEN REGISTER FILES**

(75) Inventors: **Lihuang Li**, Shenzhen (CN); **Hui Ren**, Shenzhen (CN); **Chunyu Tian**, Shenzhen (CN)

(73) Assignee: **ZTE CORPORATION**, Shenzhen, Guangdong (CN)

(21) Appl. No.: **14/008,159**

(22) PCT Filed: **Aug. 22, 2011**

(86) PCT No.: **PCT/CN2011/078721**

§ 371 (c)(1),
(2), (4) Date: **Sep. 27, 2013**

(30) **Foreign Application Priority Data**

Apr. 7, 2011 (CN) 201110086342.6

Publication Classification

(51) **Int. Cl.**
G06F 9/30 (2006.01)

(52) **U.S. Cl.**
CPC ***G06F 9/30079*** (2013.01)
USPC **712/225**

(57) **ABSTRACT**

The present disclosure discloses a method and device for data transmission between register files. The method includes that: data in a source register file are read at a Stage i of a pipeline; and the read data are transmitted to a destination register file using an idle instruction pipeline. With the method of the present disclosure, data and mask information are transmitted using an idle instruction pipeline, without addition of extra registers for data and control information buffering, thus reducing logic consumption as well as increasing utilization of an existing functional unit.

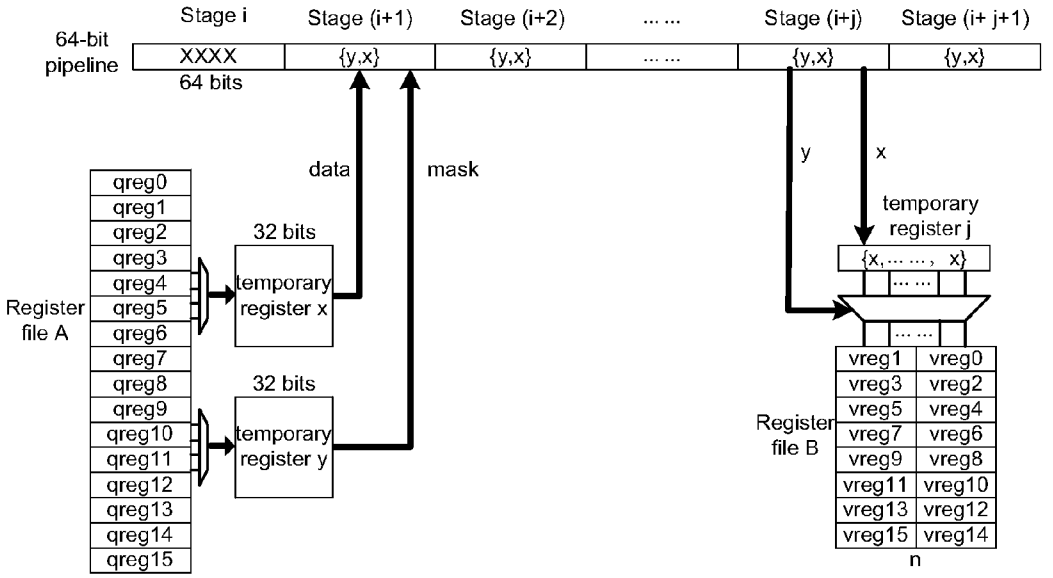


Fig. 1

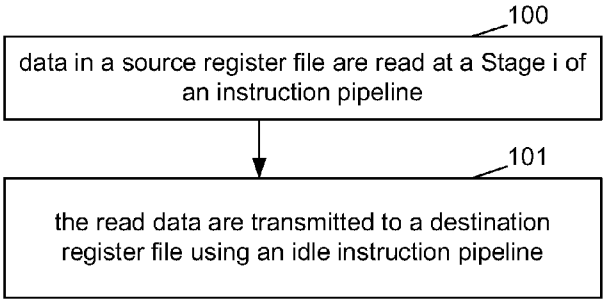


Fig. 2

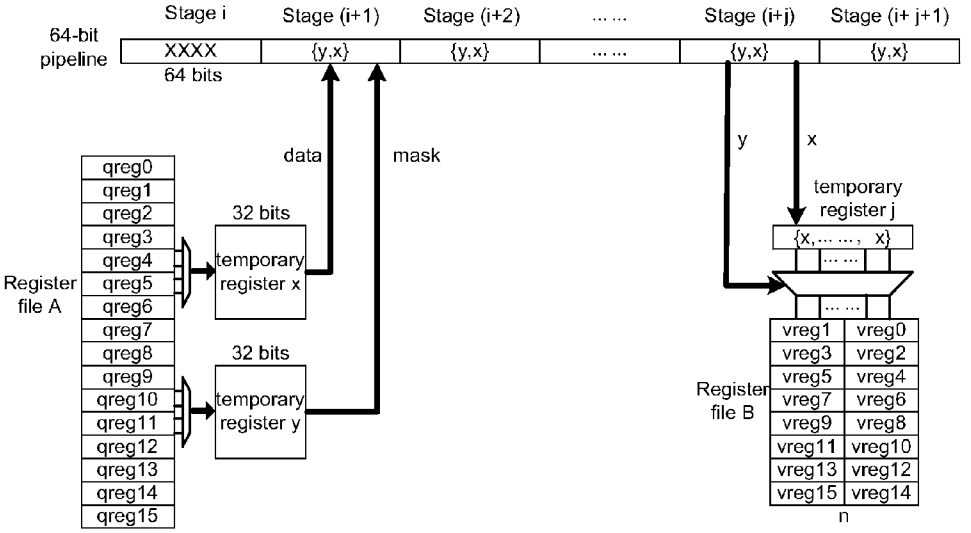
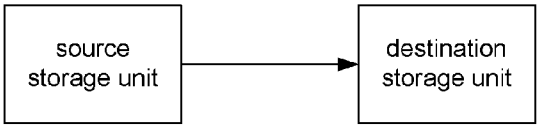


Fig. 3



METHOD AND DEVICE FOR DATA TRANSMISSION BETWEEN REGISTER FILES

TECHNICAL FIELD

[0001] The present disclosure relates to the field of data-transmission, and in particular to a method and device for data transmission between register files.

BACKGROUND

[0002] Most processors adopt a pipeline architecture. In a pipeline, there are some fixed operations in each stage, for example, reading data from a certain register file, performing calculation, and then writing a result of the calculation back into a register file. There may also be multiple register files in a processor.

[0003] In an existing integrated-circuit (IC) design, data transfer between register files are normally implemented through a data bus. Data are read from a source register file, go through relevant control logics, and are written into a destination register file through a data bus. For example, a certain processor needs to read data from register file A at the Stage i of the pipeline, and to write the data back into register file B in a Stage (i+j) of the pipeline after an instruction-pipeline delay of j stages.

[0004] During transmission of the data, transmission through a data bus requires addition of registers to buffer data and control information, which adds to resource consumption.

SUMMARY

[0005] In view of this, a main objective of the overview is to provide a method and device for data transmission between register files, which method and device are capable of reducing logic consumption and improving resource utilization.

[0006] To achieve this objective, a technical solution of the present disclosure is implemented as follows.

[0007] A method for data transmission between register files includes:

[0008] data in a source register file are read at a Stage i of an instruction pipeline; and

[0009] the read data are transmitted to a destination register file using an idle instruction pipeline.

[0010] In the above solution, the step that data in a source register file are read at a Stage i of an instruction pipeline may include:

[0011] the read data are written into a temporary register x, and a preset mask is written into a temporary register y.

[0012] In the above solution, the step that the read data are transmitted to a destination register file using an idle instruction pipeline may include:

[0013] at a Stage (i+1), data in the temporary register y and data in the temporary register x are combined into {y, x}, and the {y, x} is written into the Stage (i+1) of the idle instruction pipeline; and

[0014] the data flows stage by stage along the idle instruction pipeline into a next stage, after j clock cycles, the data are taken from the idle instruction pipeline to the destination register file at a Stage (i+j).

[0015] In the above solution, the step that the data are taken from the idle instruction pipeline to the destination register file at a Stage (i+j) may include:

[0016] a part x of the {y, x} is taken as a data unit, n copies of the data unit are combined to obtain extended data with a length equal to n times of the length of the data unit, and the extended data are written into a temporary register j; and a part y of the {y, x} is taken as a mask; and

[0017] a data unit in a section in the temporary register j corresponding to an effective bit of the mask is written into a corresponding section in the destination register file.

[0018] In the above solution, the source register file may be of 32 bits; the destination register file may be of 1024 bits; the n may be 32; and

[0019] the idle instruction pipeline may be a 64-bit instruction pipeline.

[0020] A device for data transmission between register files, includes a source storage unit and a destination storage unit, wherein

[0021] the source storage unit is configured to read data from a source register file at a Stage i of an instruction pipeline, and to transmit the read data to a destination register file using an idle instruction pipeline; and

[0022] the destination storage unit is configured to, after j clock cycles, take the data from the idle instruction pipeline to a destination register file at a Stage (i+j).

[0023] The source storage unit may be a source register file; and the destination storage unit may be a destination register file.

[0024] The source storage unit may specifically be configured to: at a Stage (i+1), combine data in a temporary register y and data in a temporary register x into {y, x}, and to write the {y, x} into the Stage (i+1) of the idle instruction pipeline, such that

[0025] the data may flow stage by stage along the idle instruction pipeline into a next stage, until after j clock cycles, the data may be taken from the idle instruction pipeline to the destination register file at the Stage (i+j).

[0026] The destination storage unit may specifically be configured to: after j clock cycles, take a part x of the {y, x} as a data unit, to combine n copies of the data unit to obtain extended data with a length equal to n times of the length of the data unit, and then to write the extended data into a temporary register j; and to take a part y of the {y, x} as a mask; and

[0027] to write a data unit in a section in the temporary register j corresponding to an effective bit of the mask into a corresponding section in the destination register file.

[0028] The source register file may be of 32 bits; the destination register file may be of 1024 bits; the n may be 32; and

[0029] the idle instruction pipeline may be a 64-bit instruction pipeline.

[0030] It can be seen from the aforementioned technical solutions provided by the present disclosure that: data in a source register file are read at a Stage i of an instruction pipeline, and the read data are transmitted to a destination register file using an idle instruction pipeline. With the solution of the present disclosure, data and mask information are transmitted using an idle instruction pipeline, without addition of extra registers for data and control information buffering, thus reducing logic consumption as well as increasing utilization of an existing functional unit.

[0031] The solutions of the present disclosure applies to processor design in a case where there are multiple instruction pipelines in the processor, and after being read from the source register file at the Stage i of an instruction pipeline,

data need to go through a clock delay of j stages before being written into the destination register file.

BRIEF DESCRIPTION OF THE DRAWINGS

[0032] FIG. **1** is a flowchart of a method for data transmission between register files according to the present disclosure;

[0033] FIG. **2** is a schematic view of transmission in an embodiment of the method for data transmission between register files according to the present disclosure; and

[0034] FIG. **3** is a schematic view of a structure of a device for data transmission between register files according to the present disclosure.

DETAILED DESCRIPTION

[0035] FIG. **1** is a flowchart of a method for data transmission between register files according to the present disclosure. As shown in FIG. **1**, the method includes the following steps.

[0036] Step **100**: data in a source register file are read at a Stage i of an instruction pipeline.

[0037] Specifically, in this step, the read data are written into a temporary register x, and a preset mask is written into a temporary register y. Using of a mask is routine technical means for a person skilled in the art, for example: when 16-bit data need to be written into a 64-bit register, 4 copies of the data may be used to form 64-bit data, then a 4-bit mask is used, that is, the copy in a section of the 64-bit data corresponding to a bit of the mask with a value of 1 (high) is written into a corresponding section in the register, which is not elaborated further here.

[0038] Step **101**: the read data are transmitted to a destination register file using an idle instruction pipeline.

[0039] In this step, at a Stage (i+1), data in the temporary register y and data in the temporary register x are combined into {y, x} which in turn is written into the Stage (i+1) of the idle instruction pipeline; then the data flow stage by stage along the idle instruction pipeline into a next stage. In the method according to the present disclosure, the idle instruction pipeline serves as a data bus. After j clock cycles, the data are taken down from the idle instruction pipeline at the Stage (i+j). The buffered x part is taken as a data unit, n copies of which are combined into extended data with a length equal to n times of the length of x, and are written into a temporary register j. The y part is taken as a mask, a data unit in a section in the temporary register j corresponding to an effective bit in the mask, for example, a bit with a value of 1, is written into a corresponding section in the directed destination register file, and correspondingly, data in a section in the destination register file corresponding to an ineffective bit of the mask remains unchanged.

[0040] In an example, n is the ratio of the size of a destination register file over that of the source register file, namely, a multiple. For example, when the source register file is of 32 bits, and the destination register file is of 1024 bits, then n=32.

[0041] With the method of the present disclosure, data and mask information are transmitted using an idle instruction pipeline, without addition of extra registers for data and control information buffering, thus reducing logic consumption as well as increasing utilization of an existing functional unit.

[0042] The method according to the present disclosure is further elaborated below with reference to embodiments.

[0043] In the embodiment, there are two register files, namely, register file A and register file B, data of one unit may be stored in a register pair in register file A, and data of n units may be stored in a register pair in register file B. In the embodiment, register file A is of 32 bits, register file B is of 1024 bits, then n=32. There are two instruction pipelines in a system, one is a 32-bit instruction pipeline, and the other is a 64-bit instruction pipeline.

[0044] In the embodiment, it is required to transmit data in register file A into register file B. That is, data in register file A are read at the Stage i of the pipeline, then after a clock delay of j stages, are written into register file B at the Stage (i+j). FIG. **2** is a schematic view of transmission in an embodiment of the method for data transmission between register files according to the present disclosure. As shown in FIG. **2**, the specific implementation includes the followings.

[0045] When a data transmission instruction is effective, source data read from register file A are written into the temporary register x at the Stage i, the read mask is written into the temporary register y. At the Stage (i+1), data in the temporary register y and data in the temporary register x are combined into {y, x}, and then are written into the idle 64-bit instruction pipeline, in which case, the 64-bit instruction pipeline serves as a data bus. After j clock cycles, at the Stage (i+j), data {y, x} are taken down from the 64-bit instruction pipeline. The x part of {y, x} is taken as a data unit, n copies of the data unit x are combined into extended data with a length equal to n times of the length of the data unit, and then the extended data are written into temporary register j. The y part in data {y, x} is taken down from the 64-bit instruction pipeline as a mask signal. In the embodiment, a data unit in a section in the temporary register j corresponding to a bit of the mask with a value of 1 (high) is written into a corresponding section of a register pair of the directed register file B. As shown in FIG. **2**, masked writing of the data is thus achieved.

[0046] FIG. **3** is a schematic view of a structure of a device for data transmission between register files according to the present disclosure. As shown in FIG. **3**, the device includes a source storage unit and a destination storage unit.

[0047] The source storage unit is configured to read data from a source register file at a Stage i of an instruction pipeline, and to transmit the read data to a destination register file using an idle instruction pipeline. The source storage unit is a source register file.

[0048] The destination storage unit is configured to, after j clock cycles, take the data from the idle instruction pipeline to the destination register file at a Stage (i+j). The destination storage unit is a destination register file.

[0049] The source storage unit may be specifically configured to: at a Stage (i+1), combine data in a temporary register y and data in a temporary register x into {y, x}, and to write the {y, x} into the Stage (i+1) of the idle instruction pipeline, such that the data flow stage by stage along the idle instruction pipeline into a next stage, after j clock cycles, the data is taken from the idle instruction pipeline to the destination register file at the Stage (i+j).

[0050] The destination storage unit may be specifically configured to: after j clock cycles, take a part x of the {y, x} as a data unit, to combine n copies of the data unit to obtain extended data with a length equal to n times of the length of the data unit, and then to write the extended data into a temporary register j; and to take a part y of the {y, x} as a mask; and to write a data unit in a section in the temporary register j corresponding to an effective bit of the mask into a corresponding section in the destination register file.

[0051] When the source register file is of 32 bits, and the destination register file is of 1024 bits, the n equals 32; and the idle instruction pipeline is a 64-bit instruction pipeline.

[0052] What described are merely preferred embodiments of the present disclosure, and are not intended to limit the scope of the present disclosure. Any modification, equivalent replacement, improvement, and the like made within the principle of the present disclosure shall be included in the scope of the present disclosure.

1. A method for data transmission between register files, comprising:

reading data from a source register file at a Stage i of an instruction pipeline; and

transmitting the read data to a destination register file using an idle instruction pipeline.

2. The method according to claim **1**, wherein the reading data from a source register file at a Stage i of an instruction pipeline comprises:

writing the read data into a temporary register x, and writing a preset mask into a temporary register y.

3. The method according to claim **2**, wherein the transmitting the read data to a destination register file using an idle instruction pipeline comprises:

at a Stage (i+1), combining data in the temporary register y and data in the temporary register x into {y, x}, and writing the {y, x} into the Stage (i+1) of the idle instruction pipeline; and

letting the data {y,x} flow stage by stage along the idle instruction pipeline into a next stage;

after j clock cycles, taking the data {y,x} from the idle instruction pipeline to the destination register file at a Stage (i+j).

4. The method according to claim **3**, wherein the taking the data {y,x} from the idle instruction pipeline to the destination register file at a Stage (i+j) comprises:

taking a part x of the {y, x} as a data unit, combining n copies of the data unit to obtain extended data with a length equal to n times of the length of the data unit, and writing the extended data into a temporary register j; and taking a part y of the {y, x} as a mask; and

writing a data unit in a section in the temporary register j corresponding to an effective bit of the mask into a corresponding section in the destination register file.

5. The method according to claim **1**, wherein the source register file is of 32 bits; the destination register file is of 1024 bits; the n is 32; and

the idle instruction pipeline is a 64-bit instruction pipeline.

6-10. (canceled)

11. The method according to claim **2**, wherein the source register file is of 32 bits; the destination register file is of 1024 bits; the n is 32; and

the idle instruction pipeline is a 64-bit instruction pipeline.

12. The method according to claim **3**, wherein the source register file is of 32 bits; the destination register file is of 1024 bits; the n is 32; and

the idle instruction pipeline is a 64-bit instruction pipeline.

13. The method according to claim **4**, wherein the source register file is of 32 bits; the destination register file is of 1024 bits; the n is 32; and

the idle instruction pipeline is a 64-bit instruction pipeline.

14. A device for data transmission between register files, comprising a reading module and a transmitting module, wherein

the reading module is configured to read data from a source register file at a Stage i of an instruction pipeline; and

the transmitting module is configured to transmit the read data to a destination register file using an idle instruction pipeline.

15. The device according to claim **14**, wherein the reading module is configured to

write the read data into a temporary register x, and write a preset mask into a temporary register y.

16. The device according to claim **15**, wherein the transmitting module is configured to:

at a Stage (i+1), combine data in the temporary register y and data in the temporary register x into {y, x}, and write the {y, x} into the Stage (i+1) of the idle instruction pipeline; and

make the data {y,x} flow stage by stage along the idle instruction pipeline into a next stage;

after j clock cycles, take the data {y,x} from the idle instruction pipeline to the destination register file at a Stage (i+j).

17. The device according to claim **16**, wherein the transmitting module is configured to

take a part x of the {y, x} as a data unit, combine n copies of the data unit to obtain extended data with a length equal to n times of the length of the data unit, and write the extended data into a temporary register j; and take a part y of the {y, x} as a mask; and

write a data unit in a section in the temporary register j corresponding to an effective bit of the mask into a corresponding section in the destination register file.

18. The device according to claim **14**, wherein the source register file is of 32 bits; the destination register file is of 1024 bits; the n is 32; and

the idle instruction pipeline is a 64-bit instruction pipeline.

* * * * *