METHOD AND DEVICE FOR DATA TRANSMISSION BETWEEN REGISTER FILES

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

Assignee: ZTE CORPORATION, Shenzhen, Guangdong (CN)

Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 544 days.

Appl. No.: 14/008,159
PCT Filed: Aug. 22, 2011
PCT No.: PCT/CN2011/078721
§ 371 (c)(1), (2), (4) Date: Sep. 27, 2013
PCT Pub. No.: WO2012/136037
PCT Pub. Date: Oct. 11, 2012

Prior Publication Data

Foreign Application Priority Data
Apr. 7, 2011 (CN) .................. 2011 1 0086342

Int. Cl.
G06F 9/32 (2006.01)
G06F 9/30 (2006.01)
G06F 9/38 (2006.01)

CPC ........ G06F 9/30079 (2013.01); G06F 9/3013 (2013.01); G06F 9/30032 (2013.01); G06F 9/3867 (2013.01)

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.

4 Claims, 1 Drawing Sheet
## References Cited

### U.S. PATENT DOCUMENTS

<table>
<thead>
<tr>
<th>Patent Number</th>
<th>Date</th>
<th>Inventor(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6,505,290</td>
<td>B1</td>
<td>Moyer et al.</td>
</tr>
<tr>
<td>6,889,317</td>
<td>B2</td>
<td>Sami</td>
</tr>
<tr>
<td>2002/0049894</td>
<td>A1</td>
<td>Moyer et al.</td>
</tr>
<tr>
<td>2005/0240930</td>
<td>A1</td>
<td>Amamiya et al.</td>
</tr>
<tr>
<td>2006/0095723</td>
<td>A1</td>
<td>Moyer et al.</td>
</tr>
</tbody>
</table>

### FOREIGN PATENT DOCUMENTS

<table>
<thead>
<tr>
<th>Patent Number</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP 03006768</td>
<td>A</td>
</tr>
<tr>
<td>JP 035221035</td>
<td>A</td>
</tr>
<tr>
<td>JP 2005284749</td>
<td>A</td>
</tr>
</tbody>
</table>

### OTHER PUBLICATIONS


* cited by examiner
Fig. 1

Data in a source register file are read at a Stage i of an instruction pipeline.

The read data are transmitted to a destination register file using an idle instruction pipeline.

Fig. 2

Fig. 3
METHOD AND DEVICE FOR DATA TRANSMISSION BETWEEN REGISTER FILES

TECHNICAL FIELD

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

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.

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+1) of the pipeline after an instruction-pipeline delay of j stages.

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

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.

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

A method for data transmission between register files includes:

- 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.

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:

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

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

- 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
- 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).

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:

- 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
- 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.

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 the idle instruction pipeline may be a 64-bit instruction pipeline.

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

- 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
- 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).

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

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
- 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).

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
- 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.

The source register file may be of 32 bits; the destination register file may be of 1024 bits; the \( n \) may be 32; and the idle instruction pipeline may be a 64-bit instruction pipeline.

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.

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

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

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

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.

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

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.

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

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 the data unit, 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.

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.

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.

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.

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).

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.

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.

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.
The invention claimed is:

1. A method for data transmission between register files, comprising:
   at a Stage i of an instruction pipeline, reading, from a source register file, data and a mask for the data:
   writing the read data into a temporary register x; and
   writing the read mask into a temporary register y;
   at a Stage (i+1) of the instruction pipeline, combining data in the temporary register y and data in the temporary register x into \( y, x \), and writing the \( y, x \) into a Stage (i+1) of an idle instruction pipeline;
   letting the data \( y, x \) flow stage by stage along the idle instruction pipeline into a next stage; and
   at a Stage (i+1) of the instruction pipeline, fetching the data \( y, x \) from the idle instruction pipeline, and
   writing the fetched data into a destination register file, wherein the fetching the data \( y, x \) from the idle instruction pipeline, and writing the fetched data into a destination register file 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 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.

2. 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.

3. A device for data transmission between register files, comprising a reading module and a transmitting module, wherein
   the reading module is configured for:
   at a Stage i of an instruction pipeline, reading, from a source register file, data and a mask for the data:
   writing the read data into a temporary register x; and
   writing the read mask into a temporary register y; and
   the transmitting module is configured for:
   at a Stage (i+1) of the instruction pipeline, combining data in the temporary register y and data in the temporary register x into \( y, x \), and writing the \( y, x \) into a Stage (i+1) of an idle instruction pipeline:
   making the data \( y, x \) flow stage by stage along the idle instruction pipeline into a next stage; and
   at a Stage (i+1) of the instruction pipeline, fetching the data \( y, x \) from the idle instruction pipeline, and
   writing the fetched data into a destination register file, 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 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.

4. The device 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.

* * * * *