(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 2 696 280 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
14.11.2018 Bulletin 2018/46

(21) Application number: **11863045.8**

(22) Date of filing: **22.08.2011**

(51) Int Cl.:
G06F 9/38 *(2018.01)*

(86) International application number:
PCT/CN2011/078721

(87) International publication number:
WO 2012/136037 (11.10.2012 Gazette 2012/41)

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

VERFAHREN UND VORRICHTUNG ZUR DATENÜBERTRAGUNG ZWISCHEN REGISTERDATEIEN

PROCÉDÉ ET DISPOSITIF DE TRANSMISSION DE DONNÉES ENTRE FICHIERS DE REGISTRE

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR

(30) Priority: **07.04.2011 CN 201110086342**

(43) Date of publication of application:
12.02.2014 Bulletin 2014/07

(73) Proprietor: **ZTE Corporation**
Shenzhen, Guangdong 518057 (CN)

(72) Inventors:
- **LI, Lihuang**
  Shenzhen
  Guangdong 518057 (CN)
- **REN, Hui**
  Shenzhen
  Guangdong 518057 (CN)
- **TIAN, Chunyu**
  Shenzhen
  Guangdong 518057 (CN)

(74) Representative: **Reichert & Lindner Partnerschaft Patentanwälte**
Bismarckplatz 8
93047 Regensburg (DE)

(56) References cited:
CN-A- 1 560 729
CN-A- 101 876 892
GB-A- 2 390 700
US-A1- 2005 240 930
US-A1- 2009 307 656
US-A1- 2010 106 944

EP 2 696 280 B1

Description

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.

[0005] US20100106944 A1 discloses a data processing apparatus and method for performing rearrangement operations. The data processing apparatus has a register data store with a plurality of registers, each register storing a plurality of data elements. Processing circuitry is responsive to control signals to perform processing operations on the data elements. An instruction decoder is responsive to at least one but no more than N rearrangement instructions, where N is an odd plural number, to generate control signals to control the processing circuitry to perform a rearrangement process at least equivalent to: obtaining as source data elements the data elements stored in N registers of said register data store as identified by the at least one re-arrangement instruction; performing a rearrangement operation to rearrange the source data elements between a regular N-way interleaved order and a de-interleaved order in order to produce a sequence of result data elements; and outputting the sequence of result data elements for storing in the register data store. This provides a particularly efficient technique for performing N-way interleave and de-interleave operations, where N is an odd number, resulting in high performance, low energy consumption, and reduced register use when compared with known prior art techniques.

SUMMARY

[0006] 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.

[0007] The features of the method and device according to the present disclosure are defined in the independent claims, and the preferable features according to the present invention are defined in the dependent claims.

[0008] 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.

[0009] 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

[0010]

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

[0011] 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.

[0012] Step 100: data in a source register file are read at a Stage i of an instruction pipeline.

[0013] 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.

[0014] Step 101: the read data are transmitted to a destination register file using an idle instruction pipeline.

[0015] 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.

[0016] 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.

[0017] 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.

[0018] The method according to the present disclosure is further elaborated below with reference to embodiments.

[0019] 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.

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.

[0020] 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.

[0021] 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.

[0022] 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.

[0023] 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.

[0024] 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).

[0025] 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.

[0026] 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.

[0027] 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.

Claims

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

reading data from a source register file at a Stage i of an instruction pipeline (100); and
transmitting the read data to a destination register file using an idle instruction pipeline (101);
wherein the reading data from a source register file at a Stage i of an instruction pipeline (100) comprises:
writing the read data into a temporary register x, and writing a preset mask into a temporary register y;
wherein the transmitting the read data to a destination register file using an idle instruction pipeline (101) 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 (101); and
letting the data {y,x} flow stage by stage along the idle instruction pipeline (101) into a next stage, wherein the idle instruction pipeline (101) serves as a data bus;
after j clock cycles, taking the data {y,x} from the idle instruction pipeline (101) to the destination register file at a Stage (i+j).

2. The method according to claim 1, wherein the taking the data {y,x} from the idle instruction pipeline (101) 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.

3. The method according to any of claims 1 to 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 (101) is a 64-bit instruction pipeline.

4. A device for data transmission between register files,
characterised by
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 (100); and
the transmitting module is configured to transmit the read data to a destination register file using an idle instruction pipeline (101);
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;
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 (101); and
make the data {y,x} flow stage by stage along the idle instruction pipeline (101) into a next stage, wherein the idle instruction pipeline (101) serves as a data bus;
after j clock cycles, take the data {y,x} from the idle instruction pipeline (101) to the destination register file at a Stage (i+j).

5. The device according to claim 4, 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.

6. The device according to any of claims 4 to 5, 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 (101) is a 64-bit instruction pipeline.

Patentansprüche

1. Ein Verfahren zur Datenübertragung zwischen Registerdateien,
gekennzeichnet durch:

Lesen von Daten aus einer Quellenregisterdatei auf einer Stufe i einer Befehlspipeline (100); und
Übertragen der gelesenen Daten in eine Zielregisterdatei mittels einer freien Befehlspipeline (101);
wobei das Lesen von Daten aus einer Quellenregisterdatei auf einer Stufe i einer Befehlspipeline (100) aufweist:

Schreiben der gelesenen Daten in ein temporäres Register x und Schreiben einer voreingestellten Maske in ein temporäres Register y;
wobei das Übertragen der gelesenen Daten in eine Zielregisterdatei mittels einer freien Befehlspipeline (101) aufweist:
auf einer Stufe (i+1) Zusammenfassen von Daten im temporären Register y und Daten im temporären Register x zu {y, x} und Schreiben von {y, x} in die Stufe (i+1) der freien Befehlspipeline (101); und
Fließenlassen der Daten {y, x} Stufe für Stufe entlang der freien Befehlspipeline (101) in eine nächste Stufe, wobei die freie Befehlspipeline (101) als ein Datenbus fungiert;
nach j Taktzyklen Überführen der Daten {y, x} von der freien Befehlspipeline (101) zur Zielregisterdatei auf einer Stufe (i+j).

2. Das Verfahren nach Anspruch 1, wobei das Überführen der Daten {y, x} von der freien Befehlspipeline (101) zur Zielregisterdatei auf einer Stufe (i+j) aufweist:

Verwenden eines Teils x von {y, x} als eine Dateneinheit, Zusammenfassen von n Kopien der Dateneinheit, um erweiterte Daten mit einer Länge gleich n mal der Länge der Dateneinheit zu erhalten, und Schreiben der erweiterten Daten in ein temporäres Register j; und
Verwenden eines Teils y von {y, x} als eine Maske; und
Schreiben einer Dateneinheit in einem Abschnitt im temporären Register j entsprechend einem effektiven Bit der Maske in einen entsprechenden Abschnitt in der Zielregisterdatei.

3. Das Verfahren nach einem der Ansprüche 1 bis 2, wobei
die Quellregisterdatei 32 Bit groß ist,
die Zielregisterdatei 1024 Bit groß ist,
n gleich 32 ist und
die freie Befehlspipeline (101) eine 64-Bit-Befehlspipeline ist.

4. Eine Vorrichtung zur Datenübertragung zwischen Registerdateien, **gekennzeichnet durch**
ein Lesemodul und ein Übertragungsmodul, wobei
das Lesemodul konfiguriert ist, um Daten aus einer Quellenregisterdatei auf einer Stufe i einer Befehlspipeline (100) zu lesen; und
das Übertragungsmodul konfiguriert ist, um die gelesenen Daten mittels einer freien Befehlspipeline (101) zu einer Zielregisterdatei zu übertragen,
wobei das Lesemodul konfiguriert ist, um
die gelesenen Daten in ein temporäres Register x zu schreiben und eine voreingestellte Maske in ein temporäres Register y zu schreiben,
wobei das Übertragungsmodul konfiguriert ist, um
auf einer Stufe (i+1) Daten im temporären Register y und Daten im temporären Register x zu {y, x} zusammenzufassen und um {y, x} in die Stufe (i+1) der freien Befehlspipeline (101) zu schreiben; und
zu bewirken, dass die Daten {y, x} Stufe für Stufe entlang der freien Befehlspipeline (101) in eine nächste Stufe fließen, wobei die freie Befehlspipeline (101) als ein Datenbus fungiert;
nach j Taktzyklen die Daten {y, x} von der freien Befehlspipeline (101) zur Zielregisterdatei auf einer Stufe (i+j) zu überführen.

5. Die Vorrichtung nach Anspruch 4, wobei das Übertragungsmodul
konfiguriert ist, um
einen Teil x von {y, x} als eine Dateneinheit zu verwenden, n Kopien der Dateneinheit zusammenzufassen, um erweiterte Daten mit einer Länge gleich n mal der Länge der Dateneinheit zu erhalten, und um die erweiterten Daten in ein temporäres Register j zu schreiben; und
einen Teil y von {y, x} als eine Maske zu verwenden; und
eine Dateneinheit in einem Abschnitt im temporären Register j entsprechend einem effektiven Bit in einen entsprechenden Abschnitt in der Zielregisterdatei zu schreiben.

6. Die Vorrichtung nach einem der Ansprüche 4 bis 5, wobei
die Quellregisterdatei 32 Bit groß ist,
die Zielregisterdatei 1024 Bit groß ist,
n gleich 32 ist und
die freie Befehlspipeline (101) eine 64-Bit-Befehlspipeline ist.

Revendications

1. Un procédé de transmission de données entre fichiers de registre,
caractérisé par
la lecture de données d'un fichier de registre de source à une étape i d'un pipeline d'instruction (100); et
la transmission des données lues à un fichier de registre de destination au moyen d'un pipeline d'instruction (101);
la lecture de données d'un fichier de registre de source à une étape i d'un pipeline d'instruction (100) comprenant:
l'écriture des données lues dans un registre temporaire x, et l'écriture d'un masque prédéfini dans un registre temporaire y;
la transmission des données lues à un fichier de registre de destination au moyen d'un pipeline d'instruction (101) inoccupé comprenant:
à une étape (i+1), la combinaison des données con-

tenues dans le registre temporaire y et des données contenues dans le registre temporaire x en {y, x}, et l'écriture de {y, x} dans l'étape (i+1) du pipeline d'instruction (101) inoccupé; et
la permission du flux des données {y, x} le long du pipeline d'instruction (101) inoccupé dans une étape suivante, étape par étape, le pipeline d'instruction (101) inoccupé servant de bus de données;
après j cycles d'horloge, le transfert des données {y, x} du pipeline d'instruction (101) inoccupé au fichier de registre de destination à une étape (i+j).

2. Le procédé selon la revendication 1, dans lequel le transfert des données {y, x} du pipeline d'instruction (101) inoccupé au fichier de registre de destination à une étape (i+j) comprend:
utiliser une partie x de {y, x} comme une unité de données, combiner n copies de l'unité de données afin d'obtenir des données étendues ayant une longueur égale à n fois la longueur de l'unité de données, et écrire les données étendues dans un registre temporaire j; et
utiliser une partie y de {y, x} comme un masque; et écrire une unité de données contenue dans une section du registre temporaire j correspondant à un bit effectif du masque dans une section correspondante du fichier de registre de destination.

3. Le procédé selon l'une des revendications 1 à 2, dans lequel
le fichier de registre de source est de 32 bit;
le fichier de registre de destination est de 1024 bit;
n est 32; et
le pipeline d'instruction (101) inoccupé est un pipeline d'instruction de 64 bit.

4. Un dispositif de transmission de données entre des fichiers de registre,
caractérisé par
un module de lecture et un module de transmission,
le module de lecture étant configuré pour lire des données d'un fichier de registre de source à une étape i d'un pipeline d'instruction (100); et
le module de transmission étant configuré pour transmettre les données lues à un fichier de registre de destination au moyen d'un pipeline d'instruction (101) inoccupé;
le module de lecture étant configuré pour
écrire les données lues dans un registre temporaire x, et écrire un masque prédéfini dans un registre temporaire y;
le module de transmission étant configuré pour:
à une étape (i+1), combiner les données contenues dans le registre temporaire y et les données contenues dans le registre temporaire x en {y, x}, et écrire {y, x} dans l'étape (i+1) du pipeline d'instruction (101) inoccupé; et
permettre le flux des données {y, x} le long du pipeline d'instruction (101) inoccupé dans une étape suivante, étape par étape, le pipeline d'instruction (101) inoccupé servant de bus de données;
après j cycles d'horloge, transférer les données {y, x} du pipeline d'instruction (101) inoccupé au fichier de registre de destination à une étape (i+j).

5. Le dispositif selon la revendication 4, dans lequel le module de
transmission est configuré pour
utiliser une partie x de {y, x} comme une unité de données, combiner n copies de l'unité de données afin d'obtenir des données étendues ayant une longueur égale à n fois la longueur de l'unité de données, et écrire les données étendues dans un registre temporaire j; et
utiliser une partie y de {y, x} comme un masque; et écrire une unité de données contenue dans une section du registre temporaire j correspondant à un bit effectif du masque dans une section correspondante du fichier de registre de destination.

6. Le dispositif selon l'une des revendications 4 à 5, dans lequel
le fichier de registre de source est de 32 bit;
le fichier de registre de destination est de 1024 bit;
n est 32; et
le pipeline d'instruction (101) inoccupé est un pipeline d'instruction de 64 bit.

Fig. 1

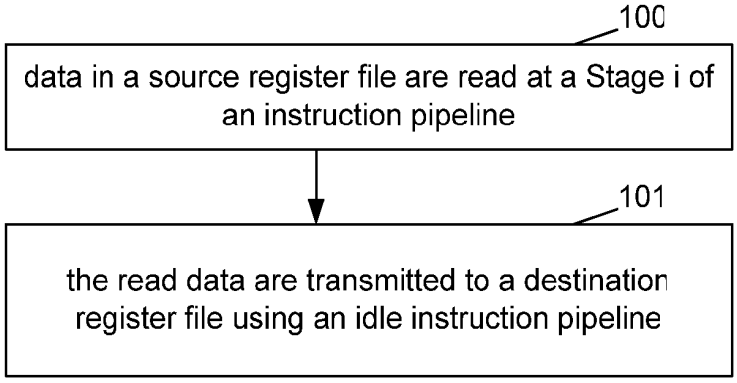


Fig. 2

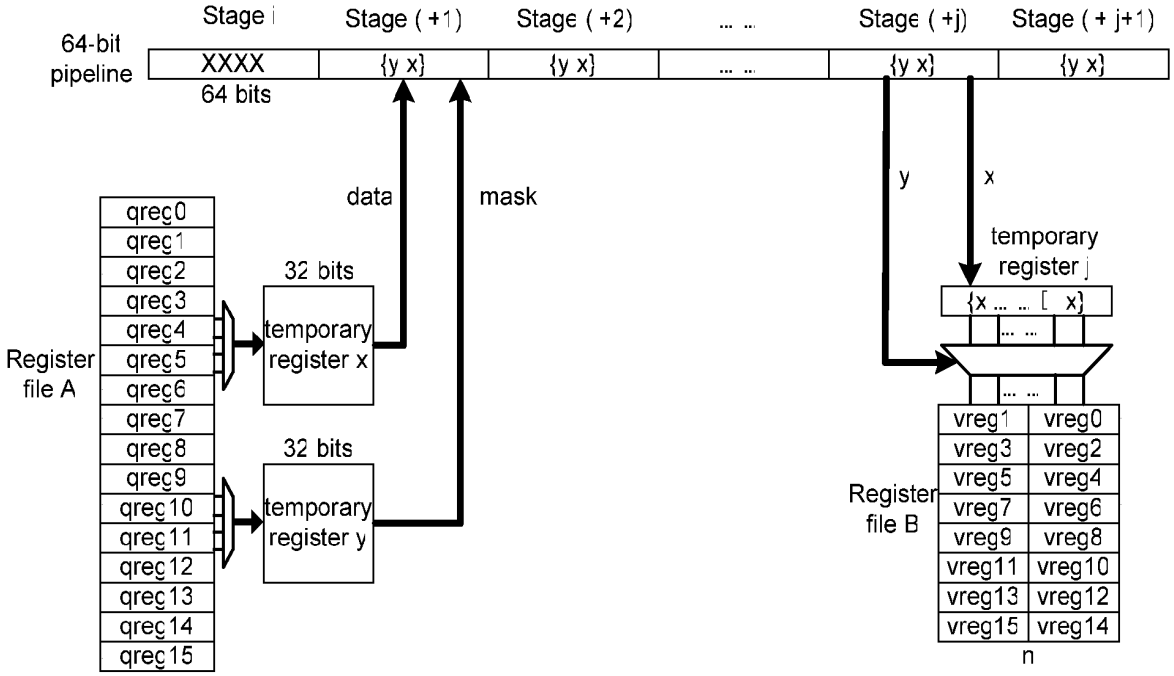
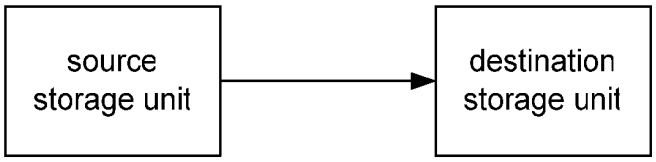


Fig. 3



REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- US 20100106944 A1 **[0005]**