Title: SYSTEM AND METHOD FOR ARBITRARY BIT PERMUTATION USING BIT-SEPARATION AND BIT-DISTRIBUTION INSTRUCTIONS

Abstract: The embodiments herein disclose a system and method for arbitrary bit permutation. According to one embodiment a method involving log_2(n) instances of bit distribution (BDST) instruction and in another embodiment involving log_3(n) instances of bit separation (BSEP) instructions to perform an arbitrary bit permutation on a programmable processor is disclosed. In one instance of the embodiment, the permute instruction separates selected bits to one side in order and unselected bits to the other side in reverse order and in another instance of the embodiment, the permute instruction distributes sequence of bits from one side to selected bit positions in order and sequence of bits from the other side to unselected bit positions in reverse order.

FIG. 2
**System and method for arbitrary bit permutation using bit-separation and bit-distribution instructions**

**Field of Invention**

The present invention relates to system and methods for performing arbitrary permutations of a sequence of bits in a processor system based on pre-defined permutation instruction sequences. In particular, the pre-defined permutation instruction sequences include bit separation instruction sequence for the sequence of bits in one instance and bit distribution instruction sequence for the sequence of bits in another instance.

**Background of the Invention**

Secure information processing using cryptography is becoming increasingly important. High speed computing, which is used in cryptography and baseband processing, demands high speed level signal processing. Bit permutation operation is a form of bit manipulation to rearrange bits, which is conventionally used to handle general bit-level signal processing. The need for secure information processing has increased with the increasing use of the public internet and wireless communications in e-commerce, e-business and
personal use. Typical use of the internet is not secure. Secure information processing typically includes authentication of users and host machines, confidentiality of messages sent over public networks, and assurances that messages, programs and data have not been maliciously changed. Conventional solutions have provided security functions by using different security protocols employing different cryptographic algorithms, such as public key, symmetric key and hash algorithms.

Some examples of conventional signal coding that involves bit permutation are data encryption standard, and bit puncturing and interleaving in forward error correction. Bit puncturing is implemented with a number of shifts, bitwise AND and bitwise OR operations.

General bit permutation would require a large number of operations and associated steps. In order to perform a general 128-bit permutation, a sequence of $4 \times 128 = 512$ operations are generally required. That would take up a lot of resource for applications such as wireless broadband baseband processing.

Some other conventional techniques have used table lookup methods to implement fixed permutations. To achieve a fixed permutation of $n$ input bits with one table lookup, a table
with $2^n$ entries is used with each entry being $n$ bits. For a
64-bit permutation, this type of table lookup would use
$2^{67}$ bytes, which is clearly infeasible. Alternatively, the
table can be broken up into smaller tables, and several
table lookup operations could be used. For example, a 64-bit
permutation could be implemented by permuting 8 consecutive
bits at a time, then combining these 8 intermediate
permutations into a final permutation. This technique,
however, suffers from requiring special hardware to do a
fast and full large table lookups and in some cases the
performance may significantly deteriorate because of
intermittent cache misses while performing large table
lookups.

A system and method for performing bit permutation by using
bit separation instructions for solving permutation problems
in cryptography, multimedia is disclosed in U.S patent
number US 7,174,014 B2 issued to Lee. Permutation
instructions according to this invention can be employed for
arbitrary permutation of bits. However, this invention
employs a general register processor (GPR) instruction set
to perform permutations, which divide the source bits into
two groups depending on configuration bits and in order to
get the result of one GRP instruction the pair of groups are
concatenated after the final permutation and this method
does not follow any bit reversal instructions. In addition,
the arbitrary permutation of the source sequence bits is done as monotonically increasing sequences of a pair of bit groups and later merging them together to form a single group of bits rather than sorting the bits based on intrinsic property of the instructions.

In yet another implementation, the general register processor instruction is introduced by employing a data bits source register, a bit mask register, and a target register. Bits in data source register with corresponding 1-bits in the mask register are shifted to one part of register with least significant bit and those with corresponding 0-bits in the mask register are shifted to the part of register with most significant bit position of the target register. The relative positions of source bits within the two groups are retained. However, this method does not incorporate any bit reversal instructions for permutation configuration of bits.

Another prior patent US 6,952,478 B2 issued to Lee discloses bit-level permutation instructions for solving permutation problems in cryptography. Both the methods disclosed in this patent can be used for providing arbitrary permutation of bits by employing a virtual omega-flip interconnection network. This method includes transforming source sequence of bits into intermediate sequences of bits and repeating the same steps till a desired sequence of bits is obtained.
using sequences of permutation instructions in an omega-flip network, whereas the proposed invention which employs different permute processor instructions for bit permutation.

Another non-blocking, conflict free routing algorithm for input and output paths in Benes networks is disclosed in IEEE paper "Matrix-based Nonblocking Routing Algorithm for Benes Networks". In this, arbitrary permutation of the connection between input and output ports is performed to determine conflict free paths for each and every input and output request. However this method is solely applied to a complete Benes network to determine the routing tags and the method to find control tags involve conflict resolution steps which may require revisiting previously selected control tags.

The present disclosure presents system and method of performing arbitrary bit permutation using a sequence of \( \log_2(n) \) different permute processor instruction. The permute processor instructions include bit separation instructions (BSEP), which separates selected bits to one side in order and separates unselected bits to the other side in reverse order; and bit distribution instructions (BDST), which distributes consecutive bits on one side to selected bit positions in order and distributes the rest of the sequence
of bits from the other side to unselected bit positions in reverse source bit order in minimal number of steps.

Summary of the Invention

System and method for arbitrary bit permutation of a plurality of n-bits is disclosed. According to one embodiment, system and method for arbitrary bit permutation using BSEP and BDST instructions disclose procedures for finding specific bit patterns in control words. The control words are generated by a sequence of permute instructions for performing a desired n-bit permutation. The control words provided by the permute instructions become parameters to a sequence of said instructions executing on specialized processor hardware configured to run based on said instructions, which can be used in solving permutation problems in cryptography, multimedia and other applications.

The system for arbitrary bit permutation of a plurality of bits in a plurality of instances comprises a permute-enhanced computing processor and a permute control words generator. According to an embodiment, the permute-enhanced computing processor of the system for arbitrary bit permutation of a plurality of bits, there is provided BDST instruction operating on two source registers and one target registers. Both the source registers and the target
registers is n-bit wide. According to the present embodiment, the plurality of n-bits is distributed to get the output in the target register. The BDST instruction distributes the sequence of bits from one part of one source register to selected bit positions in the target register in order and sequence of bits from the other part of the source register is moved to unselected bit positions in the target register in reverse order. The positions of the plurality of selected and unselected bit positions are indicated in a control word residing in the other source register. The control word is generated at each stage of the plurality of stages of bit permutation by setting bits in the control word at corresponding positions of the selected bits from an input sequence bits from the source register and clearing bits in the control word at corresponding positions of the unselected bits from the input. Furthermore, a memory subsystem associated with the permute-enhanced processor is provided with subroutine to perform arbitrary bit permutation comprising a sequence of \( \log_2(n) \) BDST instructions wherein the subroutine being parameterized by a plurality of \( \log_2(n) \) control words.

In another embodiment of the permute-enhanced processor of the system for arbitrary bit permutation of the plurality of bits, there is provided BSEP instruction operating on two source registers and one target registers. Both the source
registers and the target registers is n-bit wide. According to the present embodiment, the plurality of n-bits is separated to get the output in the target register. The BSEP instruction separates selected bits from one source register to one part of the target register in order and unselected bits from said source register to the other part of the target register in reverse order. The selected and unselected bit positions are indicated in the control word residing in the other source register. The control word is generated at each stage of a plurality of $\log_2(n)$ stages by setting bits in the control word at corresponding positions of the selected bits from the input and clearing bits in the control word at corresponding positions of the unselected bits from the input. Furthermore, the memory subsystem associated with the permute-enhanced processor is provided with subroutine to perform arbitrary bit permutation comprising a sequence of $\log_2(n)$ BSEP instructions wherein the subroutine being parameterized by $\log_2(n)$ control words.

The control words for the sequence of permute instruction to perform a desired n-bit permutation is provided by the permute control words generator. The permute control words generator generates the plurality of control words which parameterizes a sequence of $\log_2(n)$ instances of the plurality of permute instructions. The plurality of $\log_2(n)$ control words generated by the permute control words
generator in a plurality of $\log_2(n)$ stages is employed for the execution of the sequence of $\log_2(n)$ instances of the plurality of permute instructions to obtain the desired permutation of the plurality of n-bits. The sequence of instructions is done in $\log_2(n)$ stages wherein the first stage takes as input the initial permutation and the last stage generates the ordered sequence, i.e., the identity permutation. Each intervening stage of the instructions takes the intermediate permutation output from the preceding stage as input. Furthermore, each stage generates control word as a result of selecting one half of preceding permutation bits in stage-predetermined manner to one side of the resulting permutation in order and moving the other half of the unselected preceding permutation bits to the other side of the resulting permutation in reverse order.

For generating control words for a sequence of $\log_2(n)$ BDST instructions, the initial permutation of the permute control words generator is set to the desired permutation and the resulting control words sequence are applied in reverse against the BDST instructions.

For generating control words for a sequence of $\log_2(n)$ BSEP instructions, the initial permutation of the permute control words generator is set to the inverse of the desired
permutation and the resulting control words sequence are applied in order against the BSEP instructions.

**Brief Description of the Drawings**

5

Other objects, features, and advantages of the invention will be apparent from the following description when read with reference to the accompanying drawings. In the drawings, wherein like reference numerals denote corresponding parts throughout the several views:

FIG. 1 illustrates a system for performing arbitrary bit permutation comprising permute-enhanced computing processor and permute control words generator;

FIG. 2 illustrates permute-enhanced computing processor enhanced with BSEP and BDST Permute unit;

FIG. 3 illustrates a flowchart for generating control words for a sequence of BDST instructions;

FIG. 4 illustrates a flowchart for generating different control words for a sequence of BSEP instructions;

FIG. 5 exemplifies a BSEP operation in accordance with one embodiment of the present invention;
FIG. 6 exemplifies a BDST operation in accordance with one embodiment of the present invention;

FIG. 7 illustrates a permutation $P$ comprising a plurality of $n$-elements of $N$ elements.

FIG. 8 illustrates a plurality of steps for a de-permutation of 16 elements using BSEP operation;

FIG. 9 illustrates a plurality of steps to find an inverse permutation of 16 elements;

FIG. 10 exemplifies a de-permuting 16-bit inverse permutation steps; and

FIG. 11 exemplifies a permuting 16-bit configuration using BSEP operation permutation steps.

20 **Detailed Description of the Preferred Embodiments**

The present invention will now be described in detail with reference to the accompanying in drawings.

The present invention provides a system and method to perform arbitrary bit permutation using a sequence of $\log_2(n)$
bit separation (BSEP) instructions in one embodiment and a sequence of $\log_2(n)$ bit distribution (BDST) instructions in another embodiment, $n$ being the number of bits to permute.

FIG 1 illustrates the system (1103) for performing arbitrary bit permutation comprising a permute-enhanced computing processor (1101) and a permute control words generator (1102). The permute-enhanced computing processor (1101) processes a plurality permute instructions for performing arbitrary permutation of a plurality of $n$-bits. FIG. 2 illustrates permute-enhanced computing processor (1101) enhanced with BSEP and BDST permute unit. The at least one memory subsystem unit (1201) stores a plurality of data and the plurality permute instructions of a bit permute program.

At least one instruction fetch unit (1202) is employed for loading the plurality permute instructions of the program from the memory subsystem unit (1201) and a decoder unit (1203) decodes the plurality permute instructions of the program fetched by the at least one instruction fetch unit (1202). The decoder unit (1203) controls a plurality of subsequent units during execution of the plurality permute instructions of the program. At least one register file (1205) associated with the decoder unit (1203) includes a plurality of $n$-bit registers for performing as a plurality of operands during execution of the plurality permute instructions of the bit permutation program. According to
different embodiments of the present invention, the bit permutation program includes operations for either bit separation or bit distribution instructions.

The system (1103) further includes at least one load-store unit (1204) for loading the plurality of operands into the at least one register file (1205) from the at least one memory subsystem unit (1201) and for storing a plurality of resulting operands in the at least one register file (1205) to the memory subsystem unit (1201). The permute-enhanced computing processor (1101) includes a permute unit (1208) for execution of the plurality of permute instructions for performing arbitrary permutation on the plurality of operands, wherein the plurality of operands includes a source operand to permute, a control word operand for identifying a first group of the source operand bits from a second group of the source operand bits and a target operand for storing a permute instruction result. The permute-enhanced computing processor (1101) further includes at least one subroutine in the memory subsystem unit (1201). The at least one subroutine consisting of a sequence of \( \log_2(n) \) instances of the plurality of permute instructions parameterized by a plurality of control words generated by the permute control words generator (1102). The permute control words generator (1102) generates the plurality of \( \log_2(n) \) control words in a plurality of \( \log_2(n) \) stages for
execution of the sequence of $\log_2(n)$ instances of the plurality of permute instructions to obtain the desired permutation of the plurality of n-bits. A first stage among the plurality of $\log_2(n)$ stages selects an initial permutation to be an input and each subsequent stage of the plurality of $\log_2(n)$ stages generates an intermediate permutation output by selecting a plurality of predetermined bits from the input to a first side of the intermediate permutation output and by moving a sequence of unselected bits from the input to a second opposite side of the intermediate permutation output. Each stage except the first stage of the plurality of $\log_2(n)$ stages selects the intermediate permutation output from a preceding stage to be the input. The control word is generated at each stage of the plurality of $\log_2(n)$ stages by setting bits in the control word at corresponding positions of the selected bits from the input and clearing bits in the control word at corresponding positions of the unselected bits from the input.

In one embodiment, the plurality of permute instructions to the permute-enhanced computing processor (1101) for performing permutation on the plurality of operands include a BDST instruction operating on two source registers and one target registers each register being n-bit wide. The BDST instruction distributes sequence of bits from one part of
one source register to selected bit positions in the target register in order and sequence of bits from the other part of the said source register to unselected bit positions in the target register in reverse order. The positions of the plurality of selected and unselected bit positions are indicated in a control word residing in the other source register. The control word is generated at each stage of the plurality of stages of bit permutation by setting bits in the control word at corresponding positions of the selected bits from an input sequence bits from the source register and clearing bits in the control word at corresponding positions of the unselected bits from the input. Furthermore, a memory subsystem associated with the permute-enhanced processor is provided with subroutine to perform arbitrary bit permutation comprising a sequence of $\log_2(n)$ BDST instructions wherein the subroutine being parameterized by a plurality of $\log_2(n)$ control words.

In another embodiment, the plurality of permute instructions to the permute-enhanced computing processor (1101) for performing permutation on the plurality of operands include a, BSEP instruction operating on two source registers and one target registers, each register being n-bit wide. According to the present embodiment, the plurality of n-bits is separated to get the output in the target register. The BSEP instruction separates selected bits from one source
register to one part of the target register in order and
unselected bits from said source register to the other part
of the target register in reverse order. The selected and
unselected bit positions are indicated in the control word
residing in the other source register. The control word is
generated at each stage of a plurality of \( \log_2(n) \) stages by
setting bits in the control word at corresponding positions
of the selected bits from the input and clearing bits in the
control word at corresponding positions of the unselected
bits from the input. Furthermore, the memory subsystem
associated with the permute-enhanced processor is provided
with subroutine to perform arbitrary bit permutation
comprising a sequence of \( \log_2(n) \) BSEP instructions wherein
the subroutine being parameterized by \( \log_2(n) \) control words.

The permute-enhanced computing processor (1101) illustrated
in FIG. 2 is enhanced with BSEP and BDST instruction
implementation in the BSEP and BDST Permute unit (1208). In
one embodiment, the memory subsystem unit (1201) comprises
subroutine comprising sequence of \( \log_2(n) \) BDST instructions
and control word sequences for the BDST subroutine. In
another embodiment, the memory subsystem unit (1201)
comprises subroutine comprising sequence of \( \log_2(n) \) BSEP
instructions and control word sequences for the BSEP
subroutine. Furthermore, the memory subsystem unit (1201)
comprises the source data to be permuted. The load-store
unit (1204) loads source data to be permuted in addition to the control word sequence for the desired permutation into the register file (1205). Once a permute instruction in the sequence of permute instructions is loaded and decoded for execution by the instruction fetch unit (1202) and decoder unit (1203), respectively, the source register to be permuted and register with corresponding control word is loaded into the BSEP and BDST Permute unit (1208), the decoded permute instruction is executed and the resulting intermediate bit permutation is stored into the target register in the register file (1205). The target register will become the source register to be permuted for the next permute instruction in the sequence of permute instructions. The permute-enhanced computing processor (1101) includes an ALU (Arithmetic Logic Unit) (1207) common in computing processor to support the processing of other instructions like arithmetic and bit-wise operations.

FIG. 3 depicts flowchart for generating control words for a sequence of \( \log_2(n) \) BDST instructions, the initial incoming permutation (104) of the permute control words generator is set to the determined permutation (102) and the resulting control words sequence are applied in reverse against the BDST instructions (118). If a stage is not the last stage (114), the stage comprises of selecting half of the incoming permutation bits in stage-predetermined manner to one part
of the resulting permutation in order (106), moving unselected bits to the other part of said resulting permutation in reverse order (108), setting corresponding selected bits to 1 and unselected bits to 0 in the control word for the stage (110) and feeding the resulting permutation as the incoming permutation for the next stage (112). For the last stage, set bits in the control word for the last stage corresponding to first half of source bit positions in the incoming permutation to 1 and set the other bits in the control word to 0 (116).

FIG. 4 depicts flowchart for generating control words for a sequence of $\log_2(n)$ BSEP instructions, the initial incoming permutation (122) of the permute control words generator is set to the inverse of the determined permutation (120) and the resulting control words sequence are applied in order against the BSEP instructions (136). If a stage is not the last stage (132), the stage comprises selecting half of the incoming permutation bits in stage-predetermined manner to one part of the resulting permutation in order (124), moving unselected bits to the other part of said resulting permutation in reverse order (126), setting corresponding selected bits to 1 and unselected bits to 0 in the control word for the stage (128) and feeding the resulting permutation as the incoming permutation for the next stage (130). For the last stage, set bits in the control word for
the last stage corresponding to first half of source bit positions in the incoming permutation to 1 and set the other bits in the control word to 0 (134).

FIG. 5 exemplifies a BSEP operation in the permute-enhanced computing processor (1101), in accordance with one embodiment of the present invention. In this example, the BSEP operation is an 8-bit operation comprising three register operands: a source register (204), a result register (202) and a mask register (206). The bits, i.e. $b_1$, $b_3$, $b_4$ and $b_7$, of the source register (204) is distributed in normal order to one side of the result register (202) that are marked with 1-bits in the control word register (206) and the remaining bits i.e. $b_0$, $b_2$, $b_5$ and $b_6$, of the source register (204) is distributed in reverse order to the other side of the result register (202) marked with 0-bits in the control word register (206).

FIG. 6 exemplifies a BDST operation in accordance with one embodiment of the present invention. The operation is the reverse of the BSEP operation. BDST operation distributes consecutive bits in result register (202) on one side as indicated by the 1's in the mask register (206) in source bit order and the rest of the bits on the other side as indicated by the 0's in the mask register (206) in reverse source bit order.
These instructions are useful for some easily identifiable permutations. For example, bit order reversal would require a single instruction. In addition, to separate or distribute bits using the same source bit order for both groups, it may be done using two BSEP instructions or two BDST instructions, respectively. However, to be truly useful, a method from embodiments of the present invention is required by which a sequence of $\log_2(n)$ of permute instructions is used to perform an arbitrary permutation which are used extensively in coding techniques, for example, convolutional coding and convolutional turbo coding.

The control words for the sequence of permute instruction to perform a desired $n$-bit permutation is provided by the permute control words generator (1102) further comprising $\log_2(n)$ stages wherein the first stage takes as input the initial permutation and the last stage generates the ordered sequence, i.e., the identity permutation. Each intervening stage takes as input the intermediate permutation output from the preceding stage. Furthermore, each stage generates control word as a result of selecting one half of preceding permutation bits in stage-predetermined manner to one side of the resulting permutation in order and moving the other half of the unselected preceding permutation bits to the other side of the resulting permutation in reverse order.
Stage-predetermined selection for stage \( i \), except the last stage, of one half of the \( n \) bits to move to one part of the resulting permutation in order is given by the following bit positions in base-2 numbers

\[
\begin{array}{cccccc}
\log_2 n-1 & \ldots & x_{i} & 0 & 0 & x_i \ldots x_1 x_0 \\
\end{array}
\]

and

\[
\begin{array}{cccccc}
\log_2 n-1 & \ldots & x_{i} & 1 & 1 & x_i \ldots x_1 x_0 \\
\end{array}
\]

The other one half of unselected bit positions is moved to the other part of the resulting permutation in reverse order.

A method of bit permutation against some arbitrary permutation using a general 2-way split operation in \( \log_2(n) \) steps is described in FIG. 7, where \( n \) is the number of elements in the permutation. By mapping the final sequence to the ordered set of consecutive integers, a systematic method comprising steps to bring any permutation \( P \) to the ordered set, or the identity permutation \( I \), is determined based on BSEP instructions. Consequently, by going through the steps in reverse, BDST operation may be performed.

FIG. 7 illustrates a permutation \( P \) comprising of \( N \) elements. Each step consists of simple intermediate permutation wherein step-specific \( N/2 \) elements from the instructions
before are grouped into the first half of the intermediate permutation result for the step. For step 0, N/2 elements from N elements of P are grouped into the first N/2 positions. For step 1, N/4 elements are taken from the first N/2 positions and grouped into the first N/4 positions. The next N/4 elements are taken from the second N/2 positions and grouped into the second N/4 position. Similar subdivision and operation is performed until the last step. For the last step, 1 element is taken from each pair from the previous step and placed in the first N/2 positions, in order. There are $\log_2(N)$ steps. In general, for step $i$, $i = 0, \ldots, \log_2(N)-1$, for every consecutive $\left(\frac{N}{2^i}\right)$-tuple from previous step, pick half of its elements to make a new $\left(\frac{N}{2^{i+1}}\right)$-tuple. Then arrange these new $\left(\frac{1}{2}\right)\left(\frac{N}{2^{i+1}}\right)$-tuples consecutively in the first half of the resulting intermediate permutation for the step. For clarity, the steps are depicted in FIG. 5. It is applicable to any operation that may perform 2-way split in some manner. It should be clear to a person skilled in the art that the steps could be extended to using operation that performs any k-way split operation.

By going through the instructions and the methods, it is clear that a totally arbitrary permutation is changed to a very specific permutation. By labelling the elements of the resulting specific permutation using sequence of consecutive
integers of ordered set, and moving backward through the,
one would be able to find the specific elements that need to
be selected in all the way up to the original arbitrary
permutation.

Now, the above stated computation is performed with steps of
BSEP operation. First, in the last instruction, the elements
of the final permutation are labelled using the sequence (0,
1, 2,..., N-1). This sequence is also called the identity
permutation, I. Due to BSEP property wherein selected
elements are arranged into the first half in source order
and unselected elements are arranged into the second half in
reverse source order, the prior permutation would be:

$$
(\langle 0, N - 1 \rangle, \langle 1, N - 2 \rangle, \langle 2, N - 3 \rangle, \ldots, \langle \frac{N}{2} - 1, \frac{N}{2} \rangle)
$$

Each 2-tuple actually comprises an integer and its bitwise
complement. For convenience, notation $$\langle \ldots \rangle$$ is used to denote a
set of integers comprising those explicitly specified in the
angle brackets and their bitwise complements. The above
could be written as permutation,

$$
(\langle 0 \rangle, \langle 1 \rangle, \ldots, \langle \frac{N}{2} - 2 \rangle, \langle \frac{N}{2} - 1 \rangle)
$$

Now consider the second last instruction, which would
generate the intermediate permutation above. But by choosing
the complements to be the values shown at the tail of the
permutation, the permutation could also be written as:

$$
(\langle 0 \rangle, \langle 1 \rangle, \ldots, \langle \frac{N}{2} - ! \rangle, \langle \frac{N}{2} \rangle)
$$
The prior permutation that generates \((<0>, <1>, ..., <\frac{N}{2}+1>, <\frac{N}{2}>)\) over BSEP operation would be found out from following. Each pair from the first half of the resulting permutation comes from two elements in each 4-tuple from the preceding permutation. Due to BSEP property, the preceding permutation would need to be

\[
(<0\frac{N}{2}>, <i\frac{N}{2}+1>, ..., <\frac{N}{4} - (\frac{3N}{4}) - 2>, <\frac{N}{4} - 1(\frac{3N}{4}) - 1>)
\]

The above permutation is written in matrix form below for clarity and using alternate display values at the tail as:

\[
\begin{pmatrix}
0 & \frac{N}{2} \\
1 & \frac{N}{2} + 1 \\
\vdots & \vdots \\
\frac{N}{4} + 1 & \frac{3N}{4} + 1 \\
\frac{N}{4} & \frac{3N}{4}
\end{pmatrix}
\]

To gain more insight, consider the third last instruction before deducing the intermediate permutation for each step by process of induction. Again due to BSEP property, the preceding permutation would need to be:

\[
\begin{pmatrix}
0 & \frac{N}{4} & \frac{N}{2} & \frac{3N}{4} \\
1 & \frac{N}{4} + 1 & \frac{N}{2} + 1 & \frac{3N}{4} + 1 \\
\vdots & \vdots & \vdots & \vdots \\
\frac{N}{8} + 1 & \frac{3N}{8} + 1 & \frac{5N}{8} + 1 & \frac{7N}{8} + 1 \\
\frac{N}{8} & \frac{3N}{8} & \frac{5N}{8} & \frac{7N}{8}
\end{pmatrix}
\]

The table below summarizes the analysis so far and include formulation of the element selection based on the step index \(i\).
The general operation for each step is described as:

For instruction $i$, $i=0, \ldots, \log_2(n) - 2$, from prior intermediate permutation, select elements as identified below, arranged in $\frac{n}{2^{i+2}} 2^{i}$ matrix for clarity and transposed from the analysis above, using their original source positions and elements as identified by the bitwise complements of the same source positions and place the elements in the same relative order into the first half of the resulting intermediate permutation for this step.

$$
\begin{bmatrix}
0 & 1 & \cdots & 2^i - 1 \\
4 \times 2^i & 4 \times 2^i + 1 & \cdots & 5 \times 2^i - 1 \\
8 \times 2^i & 8 \times 2^i + 1 & \cdots & 9 \times 2^i - 1 \\
\vdots & \vdots & \ddots & \vdots \\
N - 4 \times 2^i & N - 4 \times 2^i + 1 & \cdots & N - 3 \times 2^i - 1
\end{bmatrix}
$$

Place the rest of the unselected elements from prior permutation into the other half of the resulting intermediate permutation in reverse relative order.
FIG. 8 to FIG. 11 illustrates, the stages in the control words generation are listed in performing arbitrary 16-bit permutation.

In stage 0, the processor picks elements \{0, 4, 8, 12, 15, 11, 7, 3\} from the original permutation into the first half of the resulting permutation preserving relative order.

In stage 1, the processor picks elements \{0, 1, 8, 9, 15, 14, 7, 6\} from the preceding permutation into the first half of the resulting permutation preserving relative order.

In stage 2, the processor picks elements \{0, 1, 2, 3, 15, 14, 13, 12\} from the preceding permutation into the first half of the resulting permutation preserving relative order.

Under stage 3, the processor picks elements \{0, 1, 2, 3, 4, 5, 6, 7\} from the preceding permutation into the first half of the resulting permutation preserving relative order. The final ordered sequence obtained is \{0, 1, 2, ..., 15\}.

FIG. 8 shows how the stages above are used to de-permute permutation \(12, 10, 3, 15, 1, 0, 9, 5, 11, 7, 4, 6, 2, 13, 8, 14\) using the prescribed method. The control word for each stage can be deduced from FIG. 8. In FIG. 8, for each stage, the highlighted elements in the permutation are assigned bit value '1' and the other elements are assigned bit value '0' in the control word. In FIG. 8, the least significant bit
(lsb) is chosen to start from the left. In programming source code, values are typically written with the lsb from the right. Once the control bits are obtained, the actual 16-bit permutation can be performed using BDST using the steps in reverse direction. The pseudo code to perform the 16-bit permutation example is shown below.

```plaintext
control [4] = {
    0b10100110_01011010,
    0b11001100_10010011,
    0b01101001_10101100,
    0b01000111_00101101
};

function ExamplePermute16(x) {
    for (i = 0 ... 3) {
        x = BDST(x, control [i]);
    }
    return x;
}
```

These control words are then used in reverse steps using BDST to perform the actual permutation.

The same method can be used to perform a permutation directly. The use of steps of BSEP can be summarized as a mapping from an arbitrary permutation $P$ to identity permutation, i.e., ordered sequence, $I$. The mapping is denoted as:

$$BSEP^k_P \rightarrow I$$

The steps of BSEP can directly be used to perform any permutation (let it be permutation $Q$), that is represented in this mapping:
Given permutation $Q$, applying its inverse permutation on permutation $Q$ would result in the identity permutation $I$, i.e., $Q^{-1}Q = I$. Multiplying $Q^{-1}$ to both side of the mapping above will get,

$$
\begin{align*}
Q^{-1} & \xrightarrow{BSEP^k} Q^{-1}Q \\
Q^{-1} & \xrightarrow{BSEP^k} I
\end{align*}
$$

To use steps of BSEP directly to perform permutation $Q$, one can apply the earlier instructions of BSEP now on the inverse of $Q$, i.e., $Q^{-1}$. The control word on every instruction can be deduced and can use the control words with BSEP instruction.

FIG. 9 shows an illustration for finding inverse permutation. Consider previous example permutation $P = (12,10,3,15,1,0,9,5,11,7,4,6,2,13,8,14)$. Finding the inverse of a permutation is straightforward. For example, the elements could be tagged with their positions and sorted using the element labels as keys. The resulting permutation of tags would be the inverse permutation. The inverse of the example permutation is therefore $P^{-1} = (5,4,12,2,10,7,11,9,14,6,1,8,0,13,15,3)$. The same method is employed by using steps of BSEP to de-permute permutation $P^{-1}$ as shown in FIG. 8.
Applying the steps and control configuration as derived from FIG. 10 on the identity permutation I as depicted in FIG. 11. It is clear that from the FIG. 11 that the resulting permutation at the end is the permutation \( P \). The pseudo code for implementing the example permutation using steps of BSEP operation is shown below.

```plaintext
control [4] = {
0b11011000_01100110,
0b00011110_01110100,
0b01011010_01011100,
0b10101001_01010101
};

function Example2Permutel6(x) {
    for (i = 0 ... 3) {
        x = BSEP(x, control [i]);
    }
    return x;
}
```

As will be readily apparent to those skilled in the art, the present invention may easily be produced in other specific forms without departing from its essential characteristics. The present embodiments is, therefore, to be considered as merely illustrative and not restrictive, the scope of the invention being indicated by the claims rather than the foregoing description, and all changes which come within therefore intended to be embraced therein.
Claims

1. A computer implemented system (1103) for performing a desired permutation of a plurality of n-bits in a plurality of instances, the computer implemented system (1103) comprising:
   a permute-enhanced computing processor (1101) for processing a plurality of permute instructions for performing arbitrary permutation of the plurality of n-bits;
   at least one memory subsystem unit (1201) for storing a plurality of data and the plurality permute instructions of a program for performing arbitrary permutation of the plurality of n-bits;
   at least one instruction fetch unit (1202) for fetching the plurality permute instructions of the program from the memory subsystem unit (1201);
   wherein the permute-enhanced computing processor (1101) further includes at least one subroutine stored in the memory subsystem unit (1201), the at least one subroutine comprising of a sequence of $\log_2(n)$ instances of the plurality of permute instructions parameterized by a plurality of control words generated by a permute control words generator (1102);
   at least one decoder unit (1203) for decoding the plurality permute instructions of the program fetched by the at least one instruction fetch unit (1202), wherein the at
at least one decoder unit (1203) controls a plurality of subsequent units during execution of the plurality permute instructions of the program;

at least one register file (1205) comprising a plurality of n-bit registers for performing a plurality of operands during execution of the plurality permute instructions of the program;

at least one load-store unit (1204) for loading the plurality of operands into the at least one register file (1205) from the at least one memory subsystem unit (1201) and for storing a plurality of resulting operands in the at least one register file (1205) to the at least one memory subsystem unit (1201);

characterised in that the permute control words generator (1102) of the computer implemented system (1103) generates the plurality of $\log_2(n)$ control words in a plurality of $\log_2(n)$ stages for execution of the sequence of $\log_2(n)$ instances of the plurality of permute instructions to obtain the desired permutation of the plurality of n-bits;

and the computer implemented system (1103) includes a permute unit (1208) for execution of the plurality of permute instructions for performing arbitrary permutation on the plurality of operands, wherein the plurality of operands includes a source operand to permute, a control word operand for identifying a first group of the source operand bits
from a second group of the source operand bits and a target operand for storing a permute instruction result.

2 The computer implemented system (1103) in claim 1 wherein the plurality of permute instructions includes a bit distribution (BDST) instruction operating on the plurality of operands to distribute a sequence of bits in relative bit order from the first group of the source operand to selected bit positions in the target operand identified by the control word operand and a sequence of bits in reverse relative bit order from the second group of the said source operand to unselected bit positions in the target operand identified by the control word operand.

3 The computer implemented system (1103) in claim 1 wherein the plurality of permute instructions includes a bit separation (BSEP) instruction operating on the plurality of operands to separate the sequence of selected bits from the source operand to one part of the target operand in relative bit order and unselected bits from said source operand to the other part of the target operand in reverse relative bit order.

4 The computer implemented system (1103) in claim 1 wherein each stage in the permute control words generator (1102) selects a plurality of predetermined bits to one part
of a permutation output in relative bit order and moves unselected bits to the other part of the permutation output in reverse relative order.

5 The computer implemented system (1103) in claim 1 wherein an initial permutation input to the first stage among the plurality of \( \log_2(n) \) stages of the permute control words generator (1102) is the desired permutation and a generated sequence of control words is applied in reverse order on a sequence of \( \log_2(n) \) instances of BDST instruction.

6 The computer implemented system (1103) in claim 1 wherein the initial permutation input to the first stage among the plurality of \( \log_2(n) \) stages of the permute control words generator (1102) is the inverse of the desired permutation and the generated sequence of control words is applied in same order on a sequence of \( \log_2(n) \) instances of BSEP instruction.

7 A method for generating a sequence of \( \log_2(n) \) control words for obtaining a desired permutation on an input of a plurality of n-bits, the method having the plurality of \( \log_2(n) \) stages comprising:

   selecting the input to be an initial permutation in a first stage among the plurality of \( \log_2(n) \) stages;
generating an intermediate permutation output in each stage of the plurality of \( \log_2(n) \) stages by selecting a plurality of predetermined bits from the input to a first side of the intermediate permutation output and by moving a plurality of unselected bits from the input to a second side of the intermediate permutation output;

selecting the intermediate permutation output from a preceding stage in each stage except the first stage of the plurality of \( \log_2(n) \) stages, the intermediate permutation output of the preceding stage is selected to be the input;

and

generating a control word at each stage of the plurality of \( \log_2(n) \) stages by setting bits in the control word at corresponding positions of the selected bits from the input and clearing bits in the control word at corresponding positions of the unselected bits from the input;

wherein the sequence of \( \log_2(n) \) control words being applied on a sequence of \( \log_2(n) \) instances of the permute instruction in the plurality of \( \log_2(n) \) stages for obtaining the desired permutation of the input having the plurality of n-bits.

8 The method in claim 7 wherein the permute instruction is a bit distribution (BDST) instruction operating on plurality of operands, the BDST instruction distributes sequence of bits from a first side of the source operand to
selected bit positions in the target operand in relative bit order and sequence of bits from a second side of said source operand to unselected bit positions in the target operand in reverse relative bit order wherein the control word operand specifies about the selected and unselected bit positions.

9 The method in claim 7 wherein the permute instruction is a bit separation (BSEP) instruction operating on plurality of operands, the BSEP instruction separates selected bits from the source operand to the first side of the target operand in relative bit order and unselected bits from said source operand to the second side of the target operand in reverse relative bit order wherein the control word operand indicates the selected and unselected bit positions.

10 The method in claim 7 wherein generating the intermediate output permutation includes a process of selecting predetermined bits to one part of the permutation output in relative bit order and moving unselected bits to the other part of the permutation output in reverse relative order.
Determine desired bit permutation from N source bit positions

Set desired bit permutation as incoming permutation for step 0

Select bits from incoming permutation identified by prescribed source bit positions and bitwise complement of said positions dependent on current step and place the selected bits in relative order in first half of outgoing permutation

NO

Next step final?

YES

Mark '1' on bit positions in control word for final step bits corresponding to first half of source bits in the incoming permutation and mark '0' on the other bit positions in said control word

Apply the determined control words on a sequence of BDST instructions in reverse, first instruction on source of N bits whereby the desired permutation of the N bits is realized after the last instruction

Place the other unselected bits from incoming permutation in reverse relative order in second half of outgoing permutation

Mark '1' on bit positions in control word for current step corresponding to selected bits from incoming permutation and mark '0' on the other bit positions in said control word

Set the outgoing permutation for the current step as the incoming permutation for the next step

FIG. 3
Determine desired bit permutation from N source bit positions

Set inverse permutation of the desired bit permutation as incoming permutation for step 0

Select bits from incoming permutation identified by prescribed source bit positions and bitwise complement of said positions dependent on current step and place the selected bits in relative order in first half of outgoing permutation

Place the other unselected bits from incoming permutation in reverse relative order in second half of outgoing permutation

Mark '1' on bit positions in control word for current step corresponding to selected bits from incoming permutation and mark '0' on the other bit positions in said control word

Set the outgoing permutation for the current step as the incoming permutation for the next step

Next step final?

Mark '1' on bit positions in control word for final step bits corresponding to first half of source bits in the incoming permutation and mark '0' on the other bit positions in said control word

Apply the determined control words, in order, on a sequence of BSEP instructions, first instruction on source of N bits whereby the desired permutation of the N bits is realized after the last instruction

FIG. 4
FIG. 7
**INTERNATIONAL SEARCH REPORT**

**International application No**  
PCT/MY2015/00Q006

---

**A. CLASSIFICATION OF SUBJECT MATTER**  
INV. G06F9/30 G06F9/76

---

**According to International Patent Classification (IPC) or to both national classification and IPC**

---

**B. FIELDS SEARCHED**

**Minimum documentation searched (classification system followed by classification symbols)**  
G06F

---

**Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched**

---

**Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)**  
EPO-Internal

---

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

<table>
<thead>
<tr>
<th>Category*</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>WO 2013/009162, AI (MIMOS BERHAD [MY]; MOHAMAD YUSRI MOHAMAD YUSOF [MY]; SMRUTI SANTOSH PA) 17 January 2013 (2013-01-17) the whole document</td>
<td>1-10</td>
</tr>
</tbody>
</table>

---

[X] Further documents are listed in the continuation of Box C.  
[X] See patent family annex.

---

"A" document defining the general state of the art which is not considered to be of particular relevance  
"E" earlier application or patent but published on or after the international filing date  
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)  
"O" document referring to an oral disclosure, use, exhibition or other means  
"P" document published prior to the international filing date but later than the priority date claimed  
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention  
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone  
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art  
"Z" document member of the same patent family

---

Date of the actual completion of the international search:  
22 June 2015

Date of mailing of the international search report:  
03/07/2015

---

Name and mailing address of the ISA/  
European Patent Office, P.B. 5818 Patentlaan 2  
NL-2280 HV Rijswijk  
Tel. (+31-70) 340-2040,  
Fax: (+31-70) 340-3016

---

Authorized officer:  
Klocke, Lynn
<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Zhijie Jerry Shi et al.: &quot;Bit permutations: Architecture, implementation, and cryptographic properties&quot;</td>
<td>1-10</td>
</tr>
<tr>
<td>Patent document cited in search report</td>
<td>Publication date</td>
<td>Patent family member(s)</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>WO 2013009162 A1</td>
<td>17-01-2013</td>
<td>NONE</td>
</tr>
<tr>
<td>WO 0189098 A2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA 2375058 A1</td>
<td></td>
<td>CA 2375058 A1</td>
</tr>
<tr>
<td>US 2002108030 A1</td>
<td></td>
<td>US 2002108030 A1</td>
</tr>
<tr>
<td>WO 0189131 A2</td>
<td></td>
<td>WO 0189131 A2</td>
</tr>
</tbody>
</table>