

RISC-V INSTRUCTION SET SHIFT INSTRUCTION IMPLEMENTATION CIRCUIT

TECHNICAL FIELD OF THE INVENTION

The invention relates to the technical field of integrated circuits, in particular to a RISC-V instruction set shift instruction realization circuit.

BACKGROUND OF THE INVENTION

Processors have been born for more than 50 years. In order to meet the diversified processing scenarios of processors and pursue higher processor performance, each processor architecture continues to expand its own instruction set, such as RISC-V instruction set excluding the basic shift instructions in base instruction set, and adding cyclic shift, forward 1 shift, funnel shift, single-bit operation and other shift type instructions to Bitmainip extension instructions. After compilation, the number of instructions executed by the program can be greatly reduced, so the processor can further achieve higher computing performance.

The RV64 instruction set contains many shift instructions, among which both the *RV BaseInteger* instruction set and the *RV Btmainip* instruction set involve shift instructions. There is no suitable logic circuit unit in the current design product to support all the above shift instructions.

In the existing products, for the high-performance design cpu, the existing barrel shifter module is easy to cause timing tension, the core frequency is limited, and can not support all shift instructions of all RISC-V architectures.

5 SUMMARY OF THE INVENTION

In view of the deficiency of the prior art, the invention discloses a RISC-V instruction set shift instruction realization circuit, which is used for solving the problem that the existing barrel shifter module is easy to cause timing tension, the core frequency is limited, and can not support all shift
10 instructions of all RISC-V architectures for the high-performance design cpu in the existing product.

The invention is realized through the following technical proposal:

The invention discloses a RISC-V instruction set shift instruction realization circuit. The circuit is integrated with a shift module. The shift
15 module is composed of a left shifter, a right shifter and a subtracter. When the instruction is emitted from the instruction buffer, it enters the decoding module to complete the instruction decoding, and the decoding information and operands enter the shift module. The subtracter is responsible for providing the shift distance Operand for the left shifter and the right shifter,
20 using the left shifter to complete the left shift instruction, the right shifter to

complete the right shift instruction, outputting the operation result, and the execution of this instruction is completed.

Further, after determining the Operand, the shift distance Operand includes *shift_amt* and *XLEN-shift_amt*. When the instruction is a single shift
 5 operation, the shifter selects the initial shift distance data *shift_am*. When the instruction is a circular shift or a funnel shift, the shift operation can be decomposed into left shift and right shift operations, and a logical OR operation is performed on the left shifter and right shifter results.

Further, the left shifter uses a logic expansion gating circuit to shape
 10 the shifted data *A*, and generates a data array *tb* composed of 64 data vectors *tb[i]*, wherein each vector *tb[i]* has data length *i*. The generation process is as follows:

S11 sets the variable *i*, the initial value is 0;

S12 takes the bit of the shifted data *A[i:0]* and assigns it to *tb[i]*;

15 S13 inverts the data *tb[i]*;

S14 the variable *i* is increased by 1, if the variable *i* is less than 64, then execute step S12 again, otherwise execute S15;

S15 outputs the data array *tb[i]*.

Furthermore, the left shifter uses the method of logical expansion to
 20 cut the shifted data *shift_decode_amt* to generate a data array *sh* composed of

64 vectors $sh[i]$, wherein the data width of the vector $sh[i]$ is i , and the specific steps are as follows:

S21 sets the variable j , the initial value of j is 0;

S22 takes the bits considered as $data[j:0]$ as $sh[j]$;

5 S23 adds 1 to the variable j , if the variable j is less than 64, execute S22 again, otherwise execute step S25;

S24 output result $sh[j]$

S25 generates a data array sh consisting of 64 vectors $sh[j]$.

Further, the left shifter performs a bitwise AND operation on the n th
 10 row vector of the tb data matrix and the sh data matrix, where n is the row index of the $tb[i]$ and $sh[j]$ data arrays. At this time, the index i is the same as the index j , and the result of the bitwise AND is logically OR processed bit by bit to obtain a 1-bit result $tmp[n]$. Since both data matrices tb and sh have 64 row vectors, the width of the final vector tmp is also 64. The process is as
 15 follows:

S31 sets the variable n , and the initial value of n is 0;

S32 takes $tb[n]$ and $sh[n]$, and performs a bitwise AND operation on $tb[n]$ and $sh[n]$ to obtain a vector tb_sh ;

S33 performs bitwise OR processing on the vector tb_sh to obtain a
 20 single-bit result $tmp[n]$;

S34, add 1 to the variable n , if the variable n is less than 64, execute step S32 again, otherwise execute S35;

S35 output result $tmp[n]$;

The n th row vector in the sh data array is subjected to bitwise logical
 5 OR operation to obtain the $sin_select[n]$ vector. According to the $sin_select[n]$ bit value, the corresponding $tmp[n]$ value or the sin value is selected as the final shift result $out[n]$, and the output data out is also 64 bits wide.

Further, the right shifter completes the right shift instruction in the same way as the left shifter completes the left shift instruction, and also uses
 10 the logic expansion strobe circuit to realize the shift function, and is the same as the shift processing unit of the left shift. only high and low inversion of input and output data is added. When the logic moves left into 1 shift, the single-bit operation instruction result is operated by the left-to-left shifter and outputs the result, and the logical right shift into 1 shift
 15 instruction result is operated by the right-to-right shifter and outputs the result. Finally, the result of the shift module selects the corresponding shifter data result according to the shift control signal.

Further, when the shift instruction is a BFP instruction, a bit region of Operand A_n is replaced with a piece of data of Operand B . after the OFF
 20 field and LEN field are extracted from the instruction code, the mask data is calculated, and the mask and replacement data are shifted to the left at the

same time. Finally, the shift results of the mask and replacement data are logically and or operated as follows:

S41 selects OFF and LEN data according to the SEL field;

S42 calculates a mask according to the LEN data, and the mask is
5 data whose bits of length LEN are all 1;

S43 uses the left shifter to shift the mask, and the shift distance is OFF data. Use the right shifter to shift the data *inb_A*, and shift the distance bit OFF data. However, the right shifter here needs to be used as a left shifter, and the high and low bit inversion operations are performed on the shift
10 data *inb_A* and the output result *rshift_outb* respectively;

S44 performs a logical AND or operation on the result;

Whrerin, the BFP instruction uses the pair pair mask mask data and data B [31:0] to logically shift to the left, and the right shifter is used as a left shifter. Before the shift data B enters the right shift data and after the result
15 output, the inversion operation is carried out, and the shift distance data of the left shifter and the right shifter are the same, both of which are OFF data. The left shifter data is input as mask data, and the right shifter data is input as *inb_A* [31:0]. The operation result of the BFP instruction is obtained by performing an OR/ AND operation on the result.

20 Further, when the shift instruction is a circular shift and a funnel shift, the output results of the left shifter and the right shifter are called at the

same time, and an OR operation is performed on the results of the two shifters;

When the shift type is cyclic left shift, the shifted data shifts n bits to the left in the left module, and the input value is zero, and the result l_res is obtained. In the right shift module, the shifted data is shifted to the right by 64murn bits, and the input value is zero, and the shift result r_res is obtained. Bitwise or processed with l_res and r_res , the result of cyclic left shift is obtained, and vice versa. The left shift module shifts 64m bits to the left, and the right shift module shifts n bits to the right. The cyclic right shift result is obtained by bitwise or operation of the shift results of the two shifters.

When the shift type is funnel shift, its operation is basically the same as the cyclic shift operation, and the shifted data of the cyclic shift data to the left and the right are the same data, while in the funnel shift operation, the left shift data and the right shift data are different data, and the data selector is responsible for data selection.

When the shift type is a single bit operation, the shifted number is 64'b1, the left shifter is used to generate mask data, and the corresponding logic operation between the operation data and the mask is completed. The single bit operation includes logic or operation, and its cyclic shift and funnel shift multiplex the operation logic in the single bit operation.

Further, it includes a processor and a memory for storing execution instructions, and the processor executes the execution instructions stored in

the memory when the RISC-V instruction set shift instruction implementation circuit is working.

The beneficial effects of the invention are:

The invention supports all the shift operation instructions contained
5 in the RISC-V architecture instruction set, and the timing is easy to converge,
and a higher main frequency can be obtained. It is especially suitable for
architecture design in pursuit of high-performance CPU.

BRIEF DESCRIPTION OF THE DRAWINGS

In order to more clearly illustrate the technical scheme in the
10 embodiment of the present invention or the prior art, the drawings that
need to be used in the embodiment or the prior art description will be
briefly introduced below. It is obvious that the drawings described below
are only some embodiments of the invention, and other drawings can be
obtained according to these drawings without creative work for those
15 skilled in the art.

Fig. 1 is a block diagram of the instruction execution completion principle of the embodiment of the present invention.

Fig. 2 is a shift module principle block diagram of an embodiment of the present invention.

Fig. 3 is a specific flow chart showing that the output data out of the embodiment of the present invention is also 64 bits wide.

Fig. 4 is a schematic diagram of an embodiment of the present invention to output a shift result value out.

5 Fig. 5 is a block diagram of the reverse operation principle of an embodiment of the present invention.

Fig. 6 is a BFP instruction format diagram of an embodiment of the present invention.

Fig. 7 is a schematic diagram of an embodiment of the present
10 invention when the shift instruction is cyclic shift and funnel shift.

Fig. 8 is a schematic diagram of the operation result output selection of the embodiment of the invention.

DETAILED DESCRIPTION OF THE EMBODIMENTS

In order to make the purpose, technical scheme and advantages of
15 the embodiment of the invention more clear, the technical scheme in the embodiment of the invention will be described clearly and completely in combination with the drawings in the embodiment of the invention. It is clear that the described embodiments are some embodiments of the present invention but not all embodiments. Based on the embodiments of the
20 invention, all other embodiments obtained by ordinary technicians in the

field without creative work fall within the scope of the protection of the invention.

Embodiment 1

The embodiment discloses a RISC-V instruction set which not only supports basic shift instructions in the base instruction set, but also supports cyclic shift, forward 1 shift, funnel shift, single bit operation and other shift type instructions in the Bitmainip extended instruction set. After compilation, the number of instructions executed by the program can be reduced by about 10%, so the processor can further achieve higher computing performance.

The RISC-V architecture supports shift instructions as follows:

Instruction Name	Shift Type	Extension Type
SLLI/SLLIW/SLL/SLLW	logical left shift operation	Base
SRLI/SRLIW/SRL/SRLW/SRAW	logical right shift operation	Base
SRAI/SRAIW/SRA	Arithmetic right shift operation	Base
SLO/SLOW/SLOI/SLOIW	Logical shift left by 1	Bitmainip
SRO/SROW/SROI/SROIW	Logical shift right by 1	Bitmainip
ROL/ROLW	Rotate left operation	Bitmainip
ROR/RORW/RORI/RORIW	Rotate right operation	Bitmainip

SBCLR/SBCLRI/SBCLRW/ SBCLRIW	Single-bit clear operation	Bitmainip
SBSET/SBSETI/SBSETW/ SBSETIW	single bit set operation	Bitmainip
SBINV/SBINVI/SBINVW/ SBINVIW	Single bit negation	Bitmainip
SBEXT/SBEXTI/SBEXTW	Single bit extraction operation	Bitmainip
FSL/FSLW/	Funnel left shift operation	Bitmainip
FSR/FSRI/FSRW/FSRIW	Funnel right shift operation	Bitmainip
SLLIU.W	Left shift operation of lower 32-bit unsigned data	Bitmainip
BFP	Bit field replacement instruction	Bitmainip

The operation of the shift instruction requires the decoding unit to provide the necessary control signal for the execution of the instruction, as shown in Fig. 1. The instruction is emitted from the instruction cache and enters the decoding module to complete the instruction decoding.

5 Subsequently, the decoding information and operands enter the shift module. After the operation is completed, the output result is submitted to the instruction, and the execution of this instruction is completed.

In this embodiment, the shift control signal required by the shift module is shown in the following table, the control signal is decoded

10 according to the instruction code, and the control signal is high and effective.

Decoded Signal	Function
is_shift	The instruction type is a shift instruction
is_rshift	right shift instruction
is_ashift	Arithmetic shift instruction
is_rwshift	The shift operation is a 32bit shift and right shift instruction
Is_oneshift	shift-in-1 instruction
is_rotate	rotate shift instruction
is_setbit	single-bit operation instructions
is_funnel	Funnel shift instructions
is_bfp	Bit field replacement instruction
is_logic_sel	Select the result of a logical operation
logic_add_type[1:0]	Type of logical operation after shift: 2'b00: bitwise inversion operation 2'b01: bitwise OR operation 2'b10: bitwise XOR operation Bitwise AND operation after bitwise inversion
shift_amt[5:0]	shift distance data

If the traditional barrel shifter is used, in order to meet the shift requirement of RV 64-bit instruction set, the shift width needs to be set to 128bit, which needs to be cascaded by a 7-stage selector, which is easy to cause timing tension.

5

The embodiment is different from the traditional barrel shifter, which is realized by logical loop expansion. The whole shift module is mainly composed of a left shifter, a right shifter and a 6bit subtracter, as shown in Fig. 2. The left shifter is responsible for completing the left shift instruction,

the right shifter is responsible for completing the right shift instruction, the left shifter and the right shifter respectively complete the shift operation required by the instruction, and the subtracter is responsible for providing shift distance operands for the left shifter and the right shifter.

5 When the operands are determined, there are only two possibilities for shifting distance operands, namely `shift_amt` and `XLEN-shift_amt` (the method takes the RV64 instruction set as an example, so `XLEN` is 64). The shift distance Operand selection of the shifter varies with different shift operations. The shift distance Operand relationship between the left shifter
10 and the right shifter is shown in the following code:

```
Assign shift_amt_sub = (is_funnel_32 | is_rolw | is_rwshift | is_bfp) ?
XLEN/2 : XLEN - shift_amt;
```

```
assign lshift_sel_amt = is_setbit | ~is_rshift;
```

```
assign rshift_sel_amt = is_rshift;
```

15 assign lshift_amt = (lshift_sel_amt | is_bfp) ? shift_amt : shift_amt_sub;

```
assign rshift_amt = (rshift_sel_amt | is_bfp) ? shift_amt : shift_amt_sub;
```

When the instruction is a single shift operation, the shifter selects the initial shift distance data, that is, *shift_amt*. When the instruction is cyclic shift or funnel shift, the shift operation can be decomposed into left shift and
20 right shift operations, and perform a logical OR operation on the results of

the left shifter and right shifter. Note that the shift distance data of left shifters and right shifters are selected differently, and the rules are entered in the following table:

	shift_amt	shift_amt_sub
Left Shifter	Rotate Left Instruction/Funnel Left Shift Instruction/Bit Area Replacement Instruction/Logical Left Shift Instruction/Single Bit Operation Instruction/Left Shift Forward 1 Instruction	Rotate Right Instructions /Funnel Right Instructions
Right Shifter	Rotate Right Shift Instruction/Funnel Right Shift Instruction/Bit Area Replacement Instruction/Arithmetic Shift/Logical Right Shift Shift/Right Shift In 1 Shift	Rotate Left Instructions /Funnel Left Instructions

Embodiment 2

5 This embodiment discloses a method for implementing a left shift unit, which is a logic expansion gate circuit. The logic expansion is to shape the shifted data A to generate a data array tb composed of 64 data vectors $tb[i]$, wherein the data length of each vector $tb[i]$ is i , and the generation process is as follows:

10 Step 1: Set the variable i , the initial value is 0;

Step 2: Take the bit of the shifted data $A[i:0]$ and assign it to $tb[i]$;

Step 3: Invert the data $tb[i]$;

Step 4: The variable i is incremented by 1. If the variable i is less than 64, perform step 2 again, otherwise, perform step 5;

Step 5: Output data array tb[i]

code show as below:

```

genvar i;

generate

5      for(i=0; i<XLEN;i++)begin:gen_rev

          assign tb[i] = inb[XLEN-1-i];

      end

endgenerate

```

Then the *shift_amt* data is decoded into one hot encoding, that is, if

10 *lshift_amt*=*i*, then *lshift_decode_amt*[*i*]=1, and the rest of the Bit bits are all 0,

and then the *lshift_decode_amt* signal is used as the logic expansion circuit

strobe signal. Perform one hot encoding on *shift_amt* according to the

following code.

```

genvar a;

15      generate

          for(a=0; a<XLEN;a=a+1)begin:one_hot

              assign lshift_decode_amt[a] = (shift_amt == a[$clog2(XLEN)-1

:0]);

```

endgenerate

Use the method of logical expansion to cut the shift data $shift_decode_amt$, and generate a data array sh composed of 64 vectors $sh[i]$, where the data width of the vector $sh[i]$ is i , and the specific steps are as follows:

Step 1: Set the initial value of variable j, j to 0;

Step 2: Take the bits considered data[$j:0$] as $sh[j]$;

Step 3: Increase the variable j by 1. If the variable j is less than 64, perform step 2 again, otherwise perform step 4;

Step 4: Output the result $sh[j]$

Generate a data array sh consisting of 64 vectors $sh[j]$, and then perform a bitwise AND operation on the n th row vector of the tb data matrix and the sh data matrix, where n is the row index of the $tb[i]$ and $sh[j]$ data arrays. At this time, the index i is the same as the index j , and the result of bitwise and is logically or processed bit by bit to get the result of 1bit $tmp[n]$. Since both the data matrices tb and sh have 64 row vectors, the width of the finally obtained vector tmp is also 64. The process is as follows:

Step 1: Set the variable n , the initial value of n is 0;

Step 2: Take $tb[n]$ and $sh[n]$, and perform bitwise AND operation on $tb[n]$ and $sh[n]$ to obtain the vector tb_sh ;

Step 3: Perform bitwise OR processing on the vector tb_sh to obtain a single-bit result $tmp[n]$;

Step 4: add 1 to variable n , if variable n is less than 64, then perform step 2 again, otherwise perform step 5;

5 Step 5: Output the result $tmp[n]$;

Perform bitwise logical OR operation on the n row vectors in the sh data array to obtain the $sin_select[n]$ vector. According to the $sin_select[n]$ bit value, the corresponding $tmp[n]$ value or the sin value is selected as the final shift result $out[n]$, and the output data out is also 64 bits wide. The specific process is shown in Fig. 3. This method is suitable for data shifting of any width. For the convenience of an example, use 16bit data as an example. Assuming that the shifted number is 0x8fe8 and the shift distance is 0x4, the data and the distance value are cut to generate a data matrix composed of 16 vectors, as shown in Fig. 4. Finally, the shift result value out is output.

15 Among them, sin is the value to be shifted in by the shift operation, and is the external input signal. In the RISC-V instruction set, the shift-in value in the shift operation is divided into zero, one, and signed bit. For different instructions, the shift-in value is different, and the selector is used to select. The selection rules are as follows.

Shift-in value	Types
0	Logical shift, cyclic shift, funnel shift, single-bit operations
1	logical shift 1 shift
data[63]	64-bit arithmetic shift
data[31]	32-bit arithmetic shift

Embodiment 3

This embodiment discusses the implementation of the right shifter and other shift instructions, which also uses the logic expansion strobe circuit to realize the shift function, and is the same as the shift processing unit of the left shifter, except that an additional inversion operation is added to the input and output data, as shown in Fig. 5. In the hardware circuit, the high-low inversion operation does not consume any logic resources.

The logical shift instruction, the arithmetic shift instruction, the shift forward 1 instruction and the single bit operation instruction all use a shifter to get the shift result. In the above types, the logic left shift, the left shift into 1 shift, the single bit operation instruction result is operated by the left shifter and outputs the result, the logic right shift, the arithmetic shift, the right shift instruction result is operated by the right shifter and the result is output. Finally, the result of the shift module can select the corresponding shifter data result according to the shift control signal.

When the instruction is a funnel shift BFP instruction, a bit region of Operand An is replaced with a piece of data of Operand B. The format of the BFP instruction is shown in Fig. 6. After the OFF field and the LEN field are

extracted from the instruction code, the mask and the replacement data need to be shifted to the left at the same time, and finally a logical AND/OR operation is performed on the shift result of the mask and the replacement data. The specific process is as follows:

- 5 1. Select OFF and LEN data according to the SEL field;
2. Calculate the mask calculation code according to the LEN data as follows:

```

genvara,i;

generate

10       for(a=0; a<XLEN;a=a+1)begin:one_hot

          assign mask_onehot[a] = (len == a[$clog2(XLEN)-1 :0]);

      endgenerate

      generate

          for(i=0;i<33;i++)begin:bfp_mask

15               assign mask_dec[i] = | mask_onehot[32:i];

          end

      endgenerate

      assign mask = mask_dec[32:1];

```

3. Use the left shifter to shift the mask, and the shift distance is OFF data. Use the right shifter to shift the data *inb_A*, and the shift distance is OFF data, but the right shifter here needs to be used as a left shifter to invert the shift data *inb_A* and the output result *rshift_outb* respectively operate;

5 4. Perform a logical AND/OR operation on the result, enter the following code:

```
assign bfp_res = (rshift_outb_rev&lshift_outb) | (inb_B& ~lshift_outb);
```

The BFP instruction needs to logically shift both the mask data and the data B [31:0]. In order to meet the needs of the two left shifters, the right
10 shifter needs to be used as the left shifter. The method is that the shift data B carries out high and low inversion operations before entering the right shift data and after the result output. The shift distance data of the left shifter and the right shifter are the same, both are OFF data, and the left shifter data is input as mask data. The data of the right shifter is inputted as *inb_A* [31:0],
15 and the operation result of the BFP instruction is obtained by performing an OR/AND operation on the result.

When the shift instruction is cyclic shift and funnel shift, the output results of the left shifter and the right shifter need to be called at the same time, and the results of the two shifters are carried out or processed, as
20 shown in Fig. 7. When the shift type is cyclic left shift, the shifted data shifts *n* bits to the left in the left module, and the input value is zero, and the result *l_res* is obtained. In the right shift module, the shifted data is shifted to 64m

bits to the right, and the input value is zero, and the shift result r_res is obtained. The cyclic left shift result is obtained by bit or processing between l_res and r_res . Circular right shift and vice versa. The left shift module shifts 64-bit to the left, and the right shift module shifts n bits to the right. The

5 cyclic right shift result is obtained by bit or operation of the two shifters.

Embodiment 4

The embodiment discloses a RISC-V instruction set shift instruction realization circuit, the shift type is funnel shift, and its operation is basically the same as the cyclic shift flow, and the difference lies in that the shifted

10 data of the cyclic shifted data is different. The shifted data of the cyclic shifted data in the left direction and the right direction are the same data, while in the funnel shift operation, the left shift data and the right shift data are different data. The data selector is responsible for data selection.

Instruction type	shifter type	shift_amt is greater than 64	shift_amt is less than 64
Funnel left	left shifter	inb_C	inb_A
	right shifter	inb_A	inb_C
Funnel right	left shifter	inb_C	inb_A
	right shifter	inb_A	inb_C

Embodiment 5

The embodiment discloses a RISC-V instruction set shift instruction realization circuit. The shift instruction is a cyclic shift, and the output results of the left shifter and the right shifter need to be called at the same time, and the results of the two shifters are carried out or processed, as shown in Fig. 7. When the shift type is cyclic left shift, the shifted data shifts n bits to the left in the left module, and the input value is zero, and the result l_res is obtained.

In the right shift module, the shifted data is shifted to $64m$ bits to the right, and the input value is zero, and the shift result r_res is obtained. The cyclic left shift operation result is obtained by bit or processing between l_res and r_res .

Circular right shift and vice versa, the left shift module shifts 64-bit to the left, and the right shift module shifts n bits to the right. The cyclic right shift operation result is obtained by bit or operation of the two shifters.

Embodiment 6

The embodiment discloses a RISC-V instruction set shift instruction realization circuit. When the shift type is a single bit operation, the number to be shifted is $64'b1$, the shift distance is the bit position distance to be operated, the left shifter is used to generate mask data, and the corresponding logic operation is carried out between the operation data and

the mask to complete the single bit operation. Single-bit operations contain logic or operations, so cyclic shift and funnel shift multiplex bitwise or operational logic in single-bit operations. The logical operations of single-bit operations are shown in the table below. The result of the operation needs to

5 be output selected, as shown in Fig. 8.

Instruction type	Type of logical operation
sbset/sbseti/sbsetw/sbsetiw	OR operation
sbclr/sbclri/sbclrw/sbclriw	AND operation, NOT operation
sbinv/sbinvi/sbinvwsbinviw	XOR
sbext/sbexti/sbextw	AND operation

Embodiment 7

The embodiment discloses a RISC-V instruction set shift instruction realization circuit. Because the output position of the final result of different shift instructions is different, it is necessary to select and output each shift

10 result. The shift operation has three outputs, namely, the left shift unit, the right shift unit and the logic operation unit. The relationship between the result output selection and instructions is as follows:

shift result output position	Included instructions
left shifter	Logical left shift instruction, logical left shift in 1 instruction
right shifter	Logical shift right instruction, logical shift right in 1 instruction, arithmetic shift instruction
logical operation unit	circular shift, funnel shift

The code is as follows:

```
assign shift_out= is_logic_sel ? shift_outb :is_rshift ? rshift_outb :
lshift_outb.
```

In summary, the invention uses two shift units to support all shift
 5 instructions of all RISC-V architectures, and proposes a logic loop expansion
 strobe method to complete the shift unit, and the invention can also be
 applied to other instruction set mechanisms. The left shifter and the right
 shifter are used to complete the operation of the cyclic shift instruction. The
 left shifter and right shifter are used to complete the funnel displacement
 10 instruction operation. The function of left shifter is realized by using logic
 loop expansion. The right shifter nests the left shifter to realize the function
 of the right shifter. It is suitable for shift instruction operation of other
 instruction sets.

All the shift operation instructions contained in the RISC-V
 15 architecture instruction set are supported, and the timing is easy to converge,
 so a higher dominant frequency can be obtained. It is especially suitable for
 architecture design in pursuit of high-performance CPU.

The above embodiments are only used to illustrate the technical
 scheme of the invention, not to limit it; although the invention is described
 20 in detail with reference to the aforementioned embodiments, ordinary
 technicians in the field should understand that they can still modify the

technical scheme recorded in the above-mentioned embodiments, or equivalent replacement of some of the technical features. These modifications or replacements do not deviate the essence of the corresponding technical scheme from the spirit and scope of the technical
5 scheme of the embodiments of the present invention.

CLAIMS

1. A RISC-V instruction set shift instruction realization circuit, which is characterized in that the circuit is integrated with a shift module, the shift module is composed of a left shifter, a right shifter and a subtracter, when the instruction is emitted from the instruction cache, it enters the decoding module to complete the instruction decoding, at this time, the decoding information and operands enter the shift module, the subtracter is responsible for providing the shift distance Operand for the left shifter and the right shifter, and the left shifter is used to complete the left shift operation instruction, the right shifter completes the right shift operation instruction, after the operation is completed, the operation result is output, and the execution of this instruction is completed.
2. The RISC-V instruction set shift instruction implementation circuit according to claim 1, wherein after determining the Operand, the shift distance Operand includes *shift_amt* and *XLEN-shift_amt*, when the instruction is a single shift operation, the shifter selects the initial shift distance data, when the instruction is cyclic shift or funnel shift, the shift operation can be divided into left shift operation and right shift operation, and the results of left shifter and right shifter can be logically or calculated.

3. The RISC-V instruction set shift instruction implementation circuit according to claim 1, wherein the left shifter uses a logic expansion strobe circuit to shape the shifted data A to generate a data array tb composed of 64 data vectors $tb[i]$, wherein the data length of each vector $tb[i]$ is i , and the generation process is as follows:

S11 sets the variable i , the initial value is 0;

S12 takes the bit of the shifted data $A[i:0]$ and assigns it to $tb[i]$;

S13 inverts the data $tb[i]$;

S14 add 1 to the variable i , if the variable i is less than 64,
execute S12 again, otherwise execute S15;

S15 outputs the data array $tb[i]$.

4. The RISC-V instruction set shift instruction implementation circuit according to claim 3, wherein the left shifter cuts the shifted data using the logic expansion method to generate a data array sh composed of 64 vector $sh[i]$, wherein the data width of the vector $sh[i]$ is i , and the specific steps are as follows:

S21 sets the variable j , the initial value of j is 0;

S22 takes the bits considered as $data[j:0]$ as $sh[j]$;

S23 adds 1 to the variable j , if the variable j is less than 64, then
execute step S22 again, otherwise execute step 5;

S24 output result $sh[j]$

S25 generates a data array sh consisting of 64 vectors $sh[j]$.

5. The RISC-V instruction set shift instruction implementation circuit according to claim 4, wherein the left shifter performs a bitwise AND operation on the n th row vector of the tb data matrix and the sh data matrix, respectively, where n is the row index of the $tb[i]$ and $sh[j]$ data arrays, at this time, the index i is the same as index j , and the result of bit by bit is logically or processed bit by bit to get the result $tmp[n]$ of 1bit, since the data matrix tb and sh both have 64 row vectors, the width of the vector tmp is 64, the process is as follows:

S31 sets the variable n , and the initial value of n is 0;

S32 takes $tb[n]$ and $sh[n]$, and performs a bitwise AND operation on $tb[n]$ and $sh[n]$ to obtain a vector tb_sh ;

S33 performs bitwise OR processing on the vector tb_sh to obtain a single-bit result $tmp[n]$;

S34 the variable n is increased by 1, if the variable n is less than 64, then execute S32 again, otherwise execute S35;

S35 output result $tmp[n]$;

the n th row vector in the sh data array is subjected to bit-wise logical OR operation to obtain the $sin_select[n]$ vector, according to the $sin_$

select[n] bit value, the corresponding *tmp[n]* value is selected as the final shift result out *[n]*, and the output data out is also 64 bits wide.

6. The RISC-V instruction set shift instruction implementation circuit according to claim 1, wherein the right shifter completes the right shift instruction in the same way as the left shifter completes the left shift instruction, it also uses the logic expansion strobe circuit to realize the shift function, and is the same as the shift processing unit of the left shifter, only adding the high and low inversion operation to the input and output data, when the logic and arithmetic shift to the left into 1 shift, the single-bit operation instruction result is operated by the left-to-left shifter and outputs the result, and the logic and arithmetic right shift into 1-shift instruction result is operated by the right shifter and outputs the result, finally, the result of the shift module selects the corresponding shifter data result according to the shift control signal.
7. The RISC-V instruction set shift instruction implementation circuit according to claim 1, wherein when the shift instruction is a BFP instruction, a bit region of Operand A_n is replaced with a piece of data from Operand B, after the OFF field and the LEN field are extracted from the script, the mask and replacement data are shifted to the left at the same time, and finally, a logical AND/OR operation is performed on the shift results of the mask and the replacement data, the specific process is as follows:

S41 selects OFF and LEN data according to the SEL field;

S42 calculates the mask based on the LEN data

S43 uses the left shifter to shift the mask, and the shift distance is OFF data. Use the right shifter to shift the data *inb_A*, and shift the distance bit OFF data, however, the right shifter here needs to be used as a left shifter to invert the shift data *inb_A* and the output result *rishift_outb* respectively;

S44 performs logical AND/OR operation on the result;

wherein, the BFP instruction uses the pair mask mask data and data B [31:0] to logically shift to the left, and the right shifter is used as a left shifter, before the shift data B enters the right shift data and after the result output, the inversion operation is carried out, and the shift distance data of the left shifter and the right shifter are the same, both of which are OFF data, the left shifter data is input as mask data, and the right shifter data is input as *inb_A* [31:0], the operation result of the BFP instruction is obtained by performing an OR/AND operation on the result.

8. The RISC-V instruction set shift instruction implementation circuit according to claim 1, wherein when the shift instruction is cyclic shift and funnel shift, the output results of the left shifter and the right

shifter are called simultaneously, and the results of the two shifters are carried out or processed;

when the shift type is cyclic left shift, the shifted data shifts n bits to the left in the left module, and the input value is zero, and the result l_res is obtained, in the right shift module, the shifted data is shifted to the right by $64murn$ bits, and the input value is zero, and the shift result r_res is obtained, bitwise or processed with l_res and r_res , the result of cyclic left shift is obtained, and vice versa, the left shift module shifts $64m$ bits to the left, and the right shift module shifts n bits to the right, the cyclic right shift result is obtained by bitwise or operation of the shift results of the two shifters.

9. The RISC-V instruction set shift instruction implementation circuit according to claim 1, wherein when the shift type is funnel shift, its operation is basically the same as the cyclic shift operation, and the shifted data of the cyclic shift data to the left and the right are the same data, while in the funnel shift operation, the left shift data and the right shift data are different data, and the data selector is responsible for data selection;

when the shift type is a single bit operation, the shifted number is $64'b1$, the left shifter is used to generate mask data, and the corresponding logic operation between the operation data and the mask is completed, the single-bit operation includes the logical OR

operation, and its cyclic shift and funnel shift multiplex the operation logic in the single-bit operation.

10. The RISC-V instruction set shift instruction realization circuit according to any one of claims 1-9, wherein the processor and a
5 memory storing execution instructions are included, when the RISC-V instruction set shift instruction implementation circuit is working, the processor executes the execution instruction stored in the memory.