

RISC-V 指令集移位指令实现电路

技术领域

本发明涉及集成电路技术领域，具体涉及一种 RISC-V 指令集移位指令实现电路。

背景技术

处理器自诞生至今已有 50 多年，在发展中为了满足处理器多样化的处理场景，追求更高的处理器性能，各个处理器架构不断扩充自身指令集，例如 RISC-V 指令集除了 base 指令集中基本移位指令，在 Bitmainip 扩展指令中增加了循环移位，进 1 移位，漏斗移位，单比特操作等移位类型指令。经过编译后，可以极大的减小程序执行的指令数量，因此处理器可以进一步获得更高的运算性能。

RV64 指令集包含众多移位指令，其中 RV BaseInteger 和 RV Btmainip 指令集均涉及移位指令，当前设计产品中并没有合适逻辑电路单元支持上述所有的移位指令。

现有产品中对于高性能设计 cpu，现有的桶形移位器模块容易造成时序紧张，核心频率受到限制，且无法支持所有 RISC-V 架构的所有移位指令。

发明内容

针对现有技术的不足，本发明公开了一种 RISC-V 指令集移位指令实现电路，用于解决现有产品中对于高性能设计 cpu，现有的桶形移位器模块容易造成时序紧张，核心频率受到限制，且无法支持所有 RISC-V 架构的所有移位指令的问题。

本发明通过以下技术方案予以实现：

本发明公开了一种 RISC-V 指令集移位指令实现电路，所述电路集成有移位模块，所述移位模块由左向移位器、右向移位器和减法器组成，当指令从指令缓存区发射出来后，进入译码模块完成指令译码，

此时译码信息以及操作数进入移位模块，减法器负责为左向移位器和右向移位器提供移位距离操作数，利用左向移位器完成左移指令，右向移位器完成右移指令，输出运算结果，本条指令执行完成。

更进一步的，确定操作数后，移位距离操作数包括 shift_amt 和 XLEN-shift_amt，当指令为单一移位操作时，移位器选择初始移位距离数据 shift_am，当指令为循环移位或漏斗移位时，移位运算可以分解为左移和右移操作，并对左向移位器和右向移位器结果进行逻辑或操作。

更进一步的，所述左向移位器采用逻辑展开选通电路对被移位数据 A 进行整形，生成 64 个数据向量 tb[i]组成的数据阵列 tb，其中每个向量 tb[i]的数据长度为 i，生成过程如下：

S11 设定变量 i，初始值为 0；

S12 取被移位数据 A[i:0]比特赋值为 tb[i]；

S13 对数据 tb[i]进行取反操作；

S14 变量 i 加 1，如果变量 i 小于 64，则重新执行步 S12，反之则执行 S15；

S15 输出数据阵列 tb[i]。

更进一步的，所述左向移位器使用逻辑展开的方法对被移位数据 shift_decode_amt 进行切割，生成一个由 64 个向量 sh[i]组成的数据阵列 sh，其中向量 sh[i]的数据宽度为 i，具体步骤如下：

S21 设定变量 j，j 的初始值位 0；

S22 取被以为数据[j:0]比特为 sh[j]；

S23 变量 j 加 1，如果变量 j 小于 64，则重新执行 S22，反之执行步骤 S25；

S24 输出结果 sh[j]

S25 生成一个由 64 个向量 sh[j]组成的数据阵列 sh。

更进一步的，所述左向移位器分别对 tb 数据矩阵与 sh 数据矩阵的第 n 个行向量进行按位与操作，其中 n 为 tb[i]和 sh[j]数据阵列的

行索引，此时索引 i 与索引 j 相同，对按位与的结果逐位逻辑或处理，得到 1bit 的结果 $tmp[n]$ ，由于数据矩阵 tb 和 sh 均有 64 个行向量，最终得到向量 tmp 的宽度也为 64，过程如下：

S31 设定变量 n ， n 的初始值为 0；

S32 取 $tb[n]$ 和 $sh[n]$ ，并对 $tb[n]$ 和 $sh[n]$ 进行按位与操作，得到向量 tb_sh ；

S33 对向量 tb_sh 进行逐位或处理，得到单比特结果 $tmp[n]$ ；

S34 变量 n 加 1，如果变量 n 小于 64，则重新执行步 S32，反之执行 S35；

S35 输出结果 $tmp[n]$ ；

其中将 sh 数据阵列中第 n 个行向量进行逐位逻辑或运算，得到 $sin_select[n]$ 向量，根据 $sin_select[n]$ 比特值选择对应 $tmp[n]$ 值或 sin 值作为最终移位结果 $out[n]$ ，输出数据 out 同样为 64 比特宽度。

更进一步的，所述右向移位器完成右移指令与所述左向移位器完成左移指令的方法相同，同样使用逻辑展开选通电路实现移位功能，且与左向移位器的移位处理单元相同，只增加对输入输出数据进行高低位反转，当逻辑左移左进 1 移位，单比特操作指令结果由左向移位器运算并输出结果，逻辑右移位右移进 1 移位指令结果由右向移位器运算并输出结果，最终移位模块的结果根据移位控制信号选择对应移位器数据结果。

更进一步的，当移位指令为 BFP 指令时，使用操作数 B 的一段数据替换操作数 A 的某个比特区域，从指令码中提取出 OFF 字段和 LEN 字段后，计算得到掩码数据，同时对掩码和替换数据进行左向移位，最后对掩码和替换数据的移位结果进行逻辑与或操作，具体过程如下：

S41 根据 SEL 字段，选择 OFF 和 LEN 数据；

S42 根据 LEN 数据计算掩码，掩码为长度为 LEN 的比特位全为 1 的数据；

S43 使用左向移位器对掩码移位，移位距离为 OFF 数据。使用右向

移位器对数据 `inb_A` 移位，移位距离位 `OFF` 数据，不过这里的右向移位器需要当作左向移位器使用，对移位数据 `inb_A` 和输出结果 `rshift_outb` 分别进行高低位反转操作；

S44 对结果进行逻辑与或操作；

其中，BFP 指令使用对掩码数据和数据 `B[31:0]` 进行逻辑左移，将右向移位器当作左向移位器使用，在移位数据 `B` 在进入右向移位数据前和结果输出后，均进行反转操作，左向移位器和右向移位器移位距离数据相同，均为 `OFF` 数据，左向移位器数据输入为掩码数据，右向移位器数据输入为 `inb_A[31:0]`，对结果进行或与操作，得到 BFP 指令运算结果。

更进一步的，当移位指令为循环移位和漏斗移位时，同时调用左向移位器和右向移位器的输出结果，并将两个移位器的结果进行或运算；

当移位类型为循环左移时，被移位数据在左向模块移向左移位 `n` 位，且移入值为零，得到结果 `l_res`，被移数据在右向移位模块中向右移位 `64-n` 位，且移入值为零，得到移位结果 `r_res`，将 `l_res` 与 `r_res` 进行按位或处理，即得到循环左移移位结果，循环右移反之亦然，左向移位模块向左移位 `64-n` 位，右向移位模块向右移位 `n` 位，对两个移位器移位结果按位或运算，得到循环右移移位结果。

当移位类型为漏斗移位时，其操作与循环移位操作相同，循环移位数据左向移位和右向移位的被移位数据为同一个数据，而漏斗移位操作中，左向移位数据和右向移位数据为不同数据，数据选择器负责进行数据选择；

当移位类型为单比特操作时，被移位数为 `64' b1`，使用左向移位器生成 `mask` 掩码，待运算数据与掩码进行对应的逻辑运算完成单比特操作，单比特操作包含了逻辑或操作，其循环移位和漏斗移位复用单比特操作中的运算逻辑。

更进一步的，包括处理器以及存储有执行指令的存储器，在 RISC-V

指令集移位指令实现电路工作时，所述处理器执行所述存储器存储的所述执行指令。

本发明的有益效果为：

本发明支持 RISC-V 架构指令集包含的所有移位操作指令，且时序容易收敛，可以得到更高的主频。尤其适合追求高性能 CPU 的架构设计。

附图说明

为了更清楚地说明本发明实施例或现有技术中的技术方案，下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本发明的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

图 1 是本发明实施例的指令执行完成原理框图；

图 2 是本发明实施例的移位模块原理框图；

图 3 是本发明实施例输出数据 out 同样为 64 比特宽度的具体流程图；

图 4 是本发明实施例输出移位结果值 out 原理图；

图 5 是本发明实施例反转操作原理框图；

图 6 是本发明实施例 BFP 指令格式图；

图 7 是本发明实施例当移位指令为循环移位和漏斗移位时的原理图；

图 8 是本发明实施例运算结果输出选择原理图。

具体实施方式

为使本发明实施例的目的、技术方案和优点更加清楚，下面将结合本发明实施例中的附图，对本发明实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例是本发明一部分实施例，而不是全部的实施例。基于本发明中的实施例，本领域普通技术人员在没有

作出创造性劳动前提下所获得的所有其他实施例，都属于本发明保护的范围。

实施例 1

本实施例公开一种 RISC-V 指令集不仅支持了 base 指令集中基本移位指令，也支持 Bitmainip 扩展指令集中的循环移位，进 1 移位，漏斗移位，单比特操作等移位类型指令。经过编译后，可以减小程序执行的指令数量约 10%，因此处理器可以进一步获得更高的运算性能。

RISC-V 架构支持移位指令如下：

指令名称	移位类型	扩展类型
SLLI/SLLIW/SLL L/ SLLW	逻辑左移运算	Base
SRLI/SRLIW/SRL L/ SRLW/SRAW	逻辑右移运算	Base
SRAI/SRAIW/SRA RA	算数右移运算	Base
SLO/SLOW/SLOI/ SLOIW	逻辑进 1 左移运算	Bitmainip
SRO/SROW/SROI/ SROIW	逻辑进 1 右移运算	Bitmainip
ROL/ROLW	循环左移运算	Bitmainip
ROR/RORW/ RORI/RORIW	循环右移运算	Bitmainip

SBCLR/SBCLRI / SBCLRW/ SBCLRIW	单比特清除运算	Bitmainip
SBSET/SBSETI / SBSETW/SBSETIW	单比特置位运算	Bitmainip
SBINV/SBINVI/ SBINVW/SBINVIW	单比特取反运算	Bitmainip
SBEXT/SBEXTI / SBEXTW	单比特提取运算	Bitmainip
FSL/ FSLW/	漏斗左移运算	Bitmainip
FSR/ FSRW/ FSRI/ FSRIW	漏斗右移运算	Bitmainip
SLLIU.W	低 32 位无符号数据左移移位运算	Bitmainip
BFP	比特区域置换指令	Bitmainip

移位指令的操作需要译码单元提供指令执行所必须的控制信号，如图 1 所示，指令从指令缓存区去发射出来，进入译码模块完成指令译码，随后译码信息以及操作数进入移位模块，运算完成后，输出结果提交指令，本条指令执行完成。

本实施例中移位模块所需要的移位控制信号见下表，控制信号根据指令码译码得到，控制信号高电平有效。

译码信号	功能
is_shift	指令类型为移位指令
is_rshift	右移操作指令
is_ashift	算数移位指令
is_rwshift	移位操作为 32bit 移位且右移指令
is_oneshift	移位进 1 指令
is_rotate	循环移位指令
is_setbit	单比特操作指令
is_funnel	漏斗移位指令
is_bfp	比特域替换指令
is_logic_sel	选择逻辑运算结果
logic_add_type[1:0]	移位后逻辑运算类型： 2'b00:按位取反操作 2'b01:按位或操作 2'b10:按位异或操作 2'b11:先按位取反后按位与操作
shift_amt[5:0]	移位距离数据

如果使用传统的桶形移位器，为满足 RV 64 位指令集移位需求，移位宽度需设置为 128bit，这就需要通过 7 级选择器进行级联实现，容易造成时序紧张。

本实施例区别于传统的桶形移位器，移位器采用逻辑循环展开方式实现，整个移位模块实现主要由一个左向移位器，一个右向移位器和一个 6bit 减法器组成，如图 2 所示。其中左向移位器负责完成左移指令，右向移位器负责完成右移指令，左向移位器和右向移位器分别完成该指令所需的移位操作，减法器负责为左向移位器和右向移位器提供移位距离操作数。

在操作数确定的情况下，移位距离操作数仅有两种可能，即 shift_amt 和 XLEN-shift_amt(方法以 RV64 指令集为例，所以 XLEN 为 64)。不同的移位操作，移位器的移位距离操作数选择会不同，左向移位器和右向移位器对应输入的移位距离操作数关系见下述代码：

```
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;
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;

```

当指令为单一移位操作时，移位器选择初始移位距离数据，即 shift_amt，当指令为循环移位或漏斗移位时，移位运算可以分解为左移和右移操作，并对左向移位器和右向移位器结果进行逻辑或操作，注意的是，左向移位器和右向移位器的移位距离数据选择不同，规则入下表：

	shift_amt	shift_amt_sub
左向移位器	循环左移指令/漏斗左移指令/比特区域置换指令/逻辑左移位指令，单比特操作指令，左移进 1 指令	循环右移指令/漏斗右移指令
右向移位器	循环右移指令/漏斗右移指令/比特区域置换指令/算数移位/逻辑右移移位/右移进 1 移位	循环左移指令/漏斗左移指令

实施例 2

本实施例公开一种左向移位单元实现方法，其为逻辑展开选通电路，逻辑展开是对被移位数据 A 进行整形，生成 64 个数据向量 tb[i] 组成的数据阵列 tb，其中每个向量 tb[i] 的数据长度为 i，生成过程如下：

- 步骤 1：设定变量 i，初始值为 0；
- 步骤 2：取被移位数据 A[i:0] 比特赋值为 tb[i]；
- 步骤 3：对数据 tb[i] 进行取反操作；
- 步骤 4：变量 i 加 1，如果变量 i 小于 64，则重新执行步骤 2，反之则执行步骤 5；

步骤 5: 输出数据阵列 tb[i]

代码如下:

```
genvar i;
generate
    for(i=0; i<XLEN;i++)begin:gen_rev
        assign tb[i] = inb[XLEN-1-i];
    end
endgenerate
```

然后将 shift_amt 数据译码变为 one hot encoding 编码, 即若 lshift_amt=i, 则 lshift_decode_amt[i+1]=1, 其余 Bit 位均为 0, 随后使用 lshift_decode_amt 信号作为逻辑展开电路选通信号。对 shift_amt 按照以下代码进行 one hot encoding 编码。

```
genvar a;
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
```

使用逻辑展开的方法对移位数据 shift_decode_amt 进行切割, 生成一个由 64 个向量 sh[i]组成的数据阵列 sh, 其中向量 sh[i]的数据宽度为 i, 具体步骤如下:

步骤 1: 设定变量 j, j 的初始值位 0;

步骤 2: 取被以为数据[j:0]比特为 sh[j];

步骤 3: 变量 j 加 1, 如果变量 j 小于 64, 则重新执行步骤 2, 反之执行步骤 4;

步骤 4: 输出结果 sh[j]

生成一个由 64 个向量 sh[j]组成的数据阵列 sh, 然后分别对 tb 数据矩阵与 sh 数据矩阵的第 n 个行向量进行按位与运算, 其中 n 为

tb[i]和 sh[j]数据阵列的行索引，此时索引 i 与索引 j 相同，对接位与的结果逐位逻辑或处理，得到 1bit 的结果 tmp[n]，由于数据矩阵 tb 和 sh 均有 64 个行向量，所以最终得到向量 tmp 的宽度也为 64。过程如下：

步骤 1：设定变量 n，n 的初始值为 0；

步骤 2：取 tb[n]和 sh[n]，并对 tb[n]和 sh[n]进行按位与操作，得到向量 tb_sh；

步骤 3：对向量 tb_sh 进行逐位或处理，得到单比特结果 tmp[n]；

步骤 4：变量 n 加 1，如果变量 n 小于 64，则重新执行步骤 2，反之执行步骤 5；

步骤 5：输出结果 tmp[n]；

将 sh 数据阵列中的 n 个行向量分别进行逐位逻辑或运算，得到 sin_select[n]向量，根据 sin_select[n]比特值选择对应 tmp[n]值或 sin 值作为最终移位结果 out[n]，输出数据 out 同样为 64 比特宽度，具体流程如图 3 所示。本方法适用于任意宽度的数据移位。为方便举例，使用 16bit 数据为例，假设被移位数 0x8fe8，移位距离 0x4，对数据和因为距离值进行切割，生成 16 个向量组成的数据矩阵，如图 4 所示，最终输出移位结果值 out。其中 sin 为移位操作需要移入的值，为外部输入信号，在 RISC-V 指令集中，移位操作中的移入值分为 zero，one，signed bit，针对不同指令移入值不同，使用选择器进行选择，选择规则如下表。

移入值	类型
0	逻辑移位，循环移位，漏斗移位，单比特操作
1	逻辑移 1 移位
data[63]	64 位算数移位
data[31]	32 位算数移位

实施例 3

本实施例论述右向移位器及其他移位指令实现，其同样使用逻辑展开选通电路实现移位功能，且与左向移位器的移位处理单元相同，

只是额外增加对输入输出数据进行反转操作，如图 5 所示。在硬件电路中，高低位反转操作不会消耗任何逻辑资源。

逻辑移位指令，算数移位指令，移位进 1 指令，单比特操作指令均使用一个移位器即可得到移位结果，上述类型中逻辑左移位，左移位进 1 移位，单比特操作指令结果由左向移位器运算并输出结果，逻辑右移位，算数移位，右移位进 1 移位指令结果由右向移位器运算并输出结果。最终移位模块的结果根据移位控制信号选择对应移位器数据结果即可。

当指令为漏斗移位 BFP 指令时，即使用操作数 B 的一段数据替换操作数 A 的某个比特区域。BFP 指令格式如图 6 所示。从指令码中提取出 OFF 字段和 LEN 字段后，需要同时对掩码和替换数据进行左向移位，最后对掩码和替换数据的移位结果进行逻辑与或操作。具体过程如下：

1. 根据 SEL 字段，选择 OFF 和 LEN 数据；
2. 根据 LEN 数据计算掩码，计算代码如下：

```
genvara, i;
generate
    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<32; i++)begin:bfp_mask
        assign mask_dec[i] = |mask_onehot[32:i];
    end
endgenerate
assign mask = mask_dec[32:1];
```

3. 使用左向移位器对掩码移位，移位距离为 OFF 数据。使用右向移位器对数据 inb_A 移位，移位距离为 OFF 数据，不过这里的右向移位器需要当作左向移位器使用，对移位数据 inb_A 和输出结果

rishift_outb 分别进行反转操作；

4. 对结果进行逻辑与或操作，入下代码：

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

BFP 指令需要同时对掩码数据和数据 B[31:0] 进行逻辑左移，为满足两个左移需要，需将右向移位器当作左向移位器使用，方法为移位数据 B 在进入右向移位数据前和结果输出后，均进行高低位反转操作，左向移位器和右向移位器移位距离数据相同，均为 OFF 数据，左向移位器数据输入为掩码数据，右向移位器数据输入为 inb_A[31:0]，对结果进行或与操作，得到 BFP 指令运算结果。

当移位指令为循环移位和漏斗移位时，需要同时调用左向移位器和右向移位器的输出结果，并将两个移位器的结果进行或处理，如图 7 所示。当移位类型为循环左移时，被移位数据在左向模块移向左移位 n 位，且移入值为零，得到结果 l_res。被移数据在右向移位模块中向右移位 64-n 位，且移入值为零，得到移位结果 r_res，将 l_res 与 r_res 进行按位或处理，即得到循环左移移位结果。循环右移反之亦然，左向移位模块向左移位 64-n 位，右向移位模块向右移位 n 位，对两个移位器移位结果按位或运算，得到循环右移移位结果。

实施例 4

本实施例公开一种 RISC-V 指令集移位指令实现电路，移位类型为漏斗移位，其操作与循环移位流程基本相同，差别在于被移位数据不同，循环移位数据左向移位和右向移位的被移位数据为同一个数据，而漏斗移位操作中，左向移位数据和右向移位数据为不同数据。数据选择器负责进行数据选择。

指令类型	移位器类型	shift_amt 大于 64	shift_amt 小于 64
漏斗左移	左向移位器	inb_C	inb_A
	右向移位器	inb_A	inb_C
漏斗右移	左向移位器	inb_C	inb_A
	右向移位器	inb_A	inb_C

实施例 5

本实施例公开一种 RISC-V 指令集移位指令实现电路，移位指令为循环移位，需要同时调用左向移位器和右向移位器的输出结果，并将两个移位器的结果进行或处理，如图 7 所示。当移位类型为循环左移时，被移位数据在左向模块移向左移位 n 位，且移入值为零，得到结果 l_res。

被移数据在右向移位模块中向右移位 64-n 位，且移入值为零，得到移位结果 r_res，将 l_res 与 r_res 进行按位或处理，即得到循环左移运算结果。

循环右移反之亦然，左向移位模块向左移位 64-n 位，右向移位模块向右移位 n 位，对两个移位器移位结果按位或运算，得到循环右移运算结果。

实施例 6

本实施例公开一种 RISC-V 指令集移位指令实现电路，移位类型为单比特操作时，被移位数为 64' b1，移位距离为待运算的比特位置距离，使用左向移位器生成掩码数据，待运算数据与掩码进行对应的逻辑运算完成单比特操作。单比特操作包含了逻辑或操作，所以循环移位和漏斗移位复用单比特操作中的按位或运算逻辑，单比特操作的逻辑运算见下表。运算结果需进行输出选择，如图 8 所示。

指令类型	逻辑运算类型
sbset/sbseti/sbsetw/sbsetiw	或运算
sbclr/sbclri/sbclrw/sbclriw	与运算，非运算
sbinv/sbinvi/sbinvwsbinviw	异或运算
sbext/sbexti/sbextw	与运算

实施例 7

本实施例公开一种 RISC-V 指令集移位指令实现电路，因不同的移位指令最终结果输出位置并不相同，所以需要将各移位结果最终进行选择输出，移位运算有三处输出，分别为左向移位单元，右向移位单元，逻辑运算单元，其中结果输出选择与指令关系如下表：

移位结果输出位置	包含指令
----------	------

左向移位器	逻辑左移指令，逻辑左移进 1 指令
右向移位器	逻辑右移指令，逻辑右移进 1 指令，算数移位指令
逻辑运算单元	循环移位，漏斗移位

代码如下：

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

综上，本发明使用两个移位单元支持所有的 RISC-V 架构的所有移位指令，提出逻辑循环展开选通的方法完成移位单元，本发明同样可应用于其他指令集机构。使用左向移位器和右向移位器完成循环移位指令的运算。使用左向移位器和右向移位器完成漏斗位移指令运算。使用逻辑循环展开，实现左向移位器功能。右向移位器嵌套左向移位器，实现右向移位器功能。适用于其他指令集移位指令运算。

支持 RISC-V 架构指令集包含的所有移位操作指令，且时序容易收敛，可以得到更高的主频。尤其适合追求高性能 CPU 的架构设计。

以上实施例仅用以说明本发明的技术方案，而非对其限制；尽管参照前述实施例对本发明进行了详细的说明，本领域的普通技术人员应当理解：其依然可以对前述各实施例所记载的技术方案进行修改，或者对其中部分技术特征进行等同替换；而这些修改或者替换，并不使相应技术方案的本质脱离本发明各实施例技术方案的精神和范围。

权 利 要 求 书

1、一种 RISC-V 指令集移位指令实现电路，其特征在于，所述电路集成有移位模块，所述移位模块由左向移位器、右向移位器和减法器组成，当指令从指令缓存区发射出来后，进入译码模块完成指令译码，此时译码信息以及操作数进入移位模块，借助减法器负责为左向移位器和右向移位器提供移位距离操作数，利用左向移位器完成左移运算指令，右向移位器完成右移运算指令，运算完成后，输出运算结果，本条指令执行完成。

2、根据权利要求 1 所述的 RISC-V 指令集移位指令实现电路，其特征在于，确定操作数后，移位距离操作数包括 shift_amt 和 XLEN-shift_amt，当指令为单一移位运算时，移位器选择初始移位距离数据，当指令为循环移位或漏斗移位时，移位运算可以分解为左移和右移运算，并对左向移位器和右向移位器结果进行逻辑或运算。

3、根据权利要求 1 所述的 RISC-V 指令集移位指令实现电路，其特征在于，所述左向移位器采用逻辑展开选通电路对被移位数据 A 进行整形，生成 64 个数据向量 tb[i]组成的数据阵列 tb，其中每个向量 tb[i]的数据长度为 i，生成过程如下：

S11 设定变量 i，初始值为 0；

S12 取被移位数据 A[i:0]比特赋值为 tb[i]；

S13 对数据 tb[i]进行取反操作；

S14 变量 i 加 1，如果变量 i 小于 64，则重新执行 S12，反之则执行 S15；

S15 输出数据阵列 tb[i]。

4、根据权利要求 3 所述的 RISC-V 指令集移位指令实现电路，其特征在于，所述左向移位器使用逻辑展开的方法对被移位数据进行切割，生成一个由 64 个向量 sh[i]组成的数据阵列 sh，其中向量 sh[i]的数据宽度为 i，具体步骤如下：

S21 设定变量 j，j 的初始值位 0；

S22 取被以为数据[j:0]比特为 sh[j]；

S23 变量 j 加 1, 如果变量 j 小于 64, 则重新执行步 S22, 反之执行步骤 5;

S24 输出结果 sh[j]

S25 生成一个由 64 个向量 sh[j]组成的数据阵列 sh。

5、根据权利要求 4 所述的 RISC-V 指令集移位指令实现电路, 其特征在于, 所述左向移位器分别对 tb 数据矩阵与 sh 数据矩阵的第 n 个行向量进行按位与操作, 其中 n 为 tb[i]和 sh[j]数据阵列的行索引, 此时索引 i 与索引 j 相同, 对按位与的结果逐位逻辑或处理, 得到 1bit 的结果 tmp[n], 由于数据矩阵 tb 和 sh 均有 64 个行向量, 最终得到向量 tmp 的宽度也为 64, 过程如下:

S31 设定变量 n, n 的初始值为 0;

S32 取 tb[n]和 sh[n], 并对 tb[n]和 sh[n]进行按位与操作, 得到向量 tb_sh;

S33 对向量 tb_sh 进行逐位或处理, 得到单比特结果 tmp[n];

S34 变量 n 加 1, 如果变量 n 小于 64, 则重新执行 S32, 反之执行 S35;

S35 输出结果 tmp[n];

其中将 sh 数据阵列中第 n 个行向量进行逐位逻辑或运算, 得到 sin_select[n]向量, 根据 sin_select[n]比特值选择对应 tmp[n]值作为最终移位结果 out[n], 输出数据 out 同样为 64 比特宽度。

6、根据权利要求 1 所述的 RISC-V 指令集移位指令实现电路, 其特征在于, 所述右向移位器完成右移指令与所述左向移位器完成左移指令的方法相同, 同样使用逻辑展开选通电路实现移位功能, 且与左向移位器的移位处理单元相同, 只增加对输入输出数据进行高低位反转操作, 当逻辑和算术左移位进 1 移位, 单比特操作指令结果由左向移位器运算并输出结果, 逻辑和算术右移位进 1 移位指令结果由右向移位器运算并输出结果, 最终移位模块的结果根据移位控制信号选择对应移位器数据结果。

7、根据权利要求1所述的RISC-V指令集移位指令实现电路，其特征在于，当移位指令为BFP指令时，使用操作数B的一段数据替换操作数A的某个比特区域，从指令码中提取出OFF字段和LEN字段后，同时对掩码和替换数据进行左向移位，最后对掩码和替换数据的移位结果进行逻辑与或操作，具体过程如下：

S41 根据SEL字段，选择OFF和LEN数据；

S42 根据LEN数据计算掩码

S43 使用左向移位器对掩码移位，移位距离为OFF数据。使用右向移位器对数据inb_A移位，移位距离为OFF数据，不过这里的右向移位器需要当作左向移位器使用，对移位数据inb_A和输出结果rshift_outb分别进行反转操作；

S44 对结果进行逻辑与或操作；

其中，BFP指令使用对掩码mask数据和数据B[31:0]进行逻辑左移，将右向移位器当作左向移位器使用，在移位数据B在进入右向移位数据前和结果输出后，均进行反转操作，左向移位器和右向移位器移位距离数据相同，均为OFF数据，左向移位器数据输入为mask数据，右向移位器数据输入为inb_A[31:0]，对结果进行或操作，得到BFP指令运算结果。

8、根据权利要求1所述的RISC-V指令集移位指令实现电路，其特征在于，当移位指令为循环移位和漏斗移位时，同时调用左向移位器和右向移位器的输出结果，并将两个移位器的结果进行或处理；

当移位类型为循环左移时，被移位数据在左向模块移向左移位n位，且移入值为零，得到结果l_res，被移数据在右向移位模块中向右移位64-n位，且移入值为零，得到移位结果r_res，将l_res与r_res进行按位或处理，即得到循环左移移位结果，循环右移反之亦然，左向移位模块向左移位64-n位，右向移位模块向右移位n位，对两个移位器移位结果按位或运算，得到循环右移移位结果。

9、根据权利要求1所述的RISC-V指令集移位指令实现电路，其

特征在于，当移位类型为漏斗移位时，其操作与循环移位操作基本相同，循环移位数据左向移位和右向移位的被移位数据为同一个数据，而漏斗移位操作中，左向移位数据和右向移位数据为不同数据，数据选择器负责进行数据选择；

当移位类型为单比特操作时，被移位数为 64' b1，使用左向移位器生成掩码数据，待运算数据与掩码进行对应的逻辑运算完成单比特操作，单比特操作包含了逻辑或操作，其循环移位和漏斗移位复用单比特操作中的运算逻辑。

10、根据权利要求 1-9 任一项所述的 RISC-V 指令集移位指令实现电路，其特征在于，包括处理器以及存储有执行指令的存储器，在如权利要求 1 至 9 中任一项所述的 RISC-V 指令集移位指令实现电路工作时，所述处理器执行所述存储器存储的所述执行指令。