1. Status寄存器模块说明：通过调用带使能端的寄存器来实现IE,EXL,ERL,IM,UM,cu3-cu0,SR,NMI位的置位；调用不带使能端的寄存器来实现未定义的位。



（1）SR、NMI中断未设计，初始化过程中将其置位未发生。

（2）由于本次设计只用到了cp0，故将cu3-cu1置位为0。

（3）当cu0\_en有效时，根据cp0不可用信号对cu0进行置位。

（4）当um\_en有效时，根据模式对um进行置位。

（5）当软件写信号有效时，用写数据中的对应比特位对IM进行置位。

（6）EXL使能端：软件写信号或者eret信号有效。

EXL的数据来源有两种，一种是写数据中对应比特位，一种是eret指令对EXL的清零操作。

根据实际情况用多路选择器进行选择。

（1）当写信号有效、eret信号无效，选择写数据中对应比特位；

（2）当写信号无效、eret信号无效，选择零；

（3）当写信号和eret信号都无效时，不能进行写入操作。

（7）ERL使能端：eret信号有效或者软件信号和cache错误同时有效，即当写信号有效cache错误信号无效时不能对ERL进行写操作。

ERL的数据来源有两种，一种是写数据中对应比特位，一种是eret指令对ERL的清零操作。

根据实际信号情况用多路选择器对数据进行选择。

1. 当写信号有效、eret信号无效，选择写数据中对应比特位；
2. 当写信号无效、eret信号无效，选择零；
3. 当写信号和eret信号都无效时，不进行写入操作。
4. IE使能端：软件写信号、流水线中EI译码信号、流水线中DI译码信号，三者中有一个信号有效时可对其进行写操作。

IE的数据来源有三种，一种是写数据对应的比特位；一种是流水线译码EI有效时的有效数据；一种是流水线译码DI有效时的无效数据。

根据实际信号情况用多路选择器对数据进行选择。

1. 当软件写信号有效、流水线译码EI无效、流水线译码DI无效时，选择写数据中对应的比特位；
2. 当软件写信号无效、流水线译码EI有效、流水线译码DI无效时，选择1；
3. 当软件写信号无效、流水线译码EI有效、流水线译码DI无效时，选择0。

Status寄存器模块信号说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 信号类型 | 数据类型 | 位宽 | 含义 |
| clk | 输入 | wire | 1-bit | 时钟信号 |
| rst | 输入 | wire | 1-bit | 寄存器复位信号 |
| r\_p | 输入 | wire | 1-bit | 流水线读信号 |
| r\_h | 输入 | wire | 1-bit | 硬件读信号 |
| ei\_en | 输入 | wire | 1-bit | 流水线指令ei译码信号 |
| di\_en | 输入 | wire | 1-bit | 流水线指令di译码信号 |
| cu\_en | 输入 | wire | 1-bit | cu0写入使能端 |
| cp0\_unusable | 输入 | wire | 1-bit | cp0不可用信号 |
| um\_en | 输入 | wire | 1-bit | um写入使能端 |
| um | 输入 | wire | 1-bit | 模式位（用户、系统） |
| eret | 输入 | wire | 1-bit | 流水线指令eret译码信号 |
| we\_s | 输入 | wire | 1-bit | 软件写信号 |
| sr | 输入 | wire | 1-bit | 软件重置位 |
| nmi | 输入 | wire | 1-bit | 不可屏蔽的中断位 |
| write\_data | 输入 | wire | 32-bit | 写数据 |
| read\_data | 输出 | wire | 32-bit | 读出的数据 |
| re | 内部信号 | wire | 1-bit | 读信号 |
| ie\_en | 内部信号 | wire | 1-bit | IE使能端 |
| ie\_data | 内部信号 | wire | 1-bit | IE数据结果 |
| exl\_en | 内部信号 | wire | 1-bit | EXL使能端 |
| erl\_en | 内部信号 | wire | 1-bit | ERL使能端 |
| exl\_data | 内部信号 | wire | 1-bit | EXL数据 |
| erl\_data | 内部信号 | wire | 1-bit | ERL数据 |
| status | 内部信号 | wire | 32-bit | status寄存器 |

2、Cause寄存器模块说明：通过调用带使能端的寄存器来实现BD,IV,IP,exception\_code位的置位；调用不带使能端的寄存器来实现未定义的位。



对于IP[1:0]的置位通过软件置位来实现，其余通过硬件置位实现。

（1）BD位的置位通过流水线传来的bd\_p置位，使能端发生异常或中断。

（2）IV位的置位当发生中断时将其置为有效，其他情况置为无效。

（3）Exception\_code当发生异常时对其进行置位，数据来源流水线。

（4）IP[7:2]当发生硬件中断时对其进行置位，即当硬件中断请求有效时，将硬件中断优先级写入对应的原因寄存器。

（5）IP[1:0]当发生软件中断时对其进行置位，即当软件中断请求有效时，将软件中断优先级写入对应的原因寄存器。

（6）当流水线读信号或者硬件读信号有效时，对寄存器进行读操作。

Cause寄存器模块信号说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 信号类型 | 数据类型 | 位宽 | 含义 |
| clk | 输入 | wire | 1-bit | 时钟信号 |
| rst | 输入 | wire | 1-bit | 寄存器复位信号 |
| r\_p | 输入 | wire | 1-bit | 流水线读信号 |
| r\_h | 输入 | wire | 1-bit | 硬件读信号 |
| bd\_p | 输入 | wire | 1-bit | 流水线BD信号 |
| exception\_abort | 输入 | wire | 1-bit | 流水线异常信号 |
| irq\_h | 输入 | wire | 1-bit | 硬件中断信号 |
| ip\_h | 输入 | wire | 6-bit | 硬件中断优先级 |
| irq\_s | 输入 | wire | 1-bit | 软件中断信号 |
| ip\_s | 输入 | wire | 2-bit | 软件中断优先级 |
| exception\_code | 输入 | wire | 5-bit | 异常编码 |
| read\_data | 输出 | wire | 32-bit | 读出的数据 |
| re | 输入 | wire | 1-bit | 读信号 |
| we\_h | 内部信号 | wire | 1-bit | 硬件写信号 |
| cause | 内部信号 | wire | 32-bit | 原因寄存器 |

1. EPC寄存器模块说明：通过调用带使能端的寄存器来实现EPC的置位。



（1）epc\_t中的数据根据流水线中BD位选择pc\_p、pc\_p - 4；当 流水线中BD位有效时，选择pc\_p - 4；无效时选择pc\_p。

（2）当we\_s有效、we\_h无效时，选择write\_data；we\_s无效、we\_h有效时，选择epc\_t中的数据；we\_s、we\_h都无效时，不进行写入操作。

EPC寄存器模块信号说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 信号类型 | 数据类型 | 位宽 | 含义 |
| clk | 输入 | wire | 1-bit | 时钟信号 |
| rst | 输入 | wire | 1-bit | 寄存器复位信号 |
| r\_p | 输入 | wire | 1-bit | 流水线读信号 |
| r\_h | 输入 | wire | 1-bit | 硬件读信号 |
| we\_h | 输入 | wire | 1-bit | 硬件写信号 |
| we\_s | 输入 | wire | 1-bit | 软件写信号 |
| bd\_p | 输入 | wire | 1-bit | 流水线BD位 |
| write\_data | 输入 | wire | 32-bit | 写数据 |
| pc\_p | 输入 | wire | 32-bit | 流水线中pc |
| read\_data | 输出 | wire | 32-bit | 读出的数据 |
| we | 内部信号 | wire | 1-bit | 写信号 |
| re | 内部信号 | wire | 1-bit | 读信号 |
| epc\_t | 内部信号 | wire | 32-bit | epc中间结果 |
| epc\_data | 内部信号 | wire | 32-bit | epc寄存器数据 |
| epc | 内部信号 | wire | 32-bit | epc寄存器 |

1. ErrorEPC寄存器模块说明：通过调用带使能端的寄存器来实现ErrorEPC的置位。



（1）err\_epc\_t中的数据根据流水线中BD位选择pc\_p、pc\_p - 4；当流水线中BD位有效时，选择pc\_p - 4；无效时选择pc\_p。

（2）当we\_s有效、we\_h无效时，选择write\_data；we\_s无效、we\_h有效时，选择err\_epc\_t中的数据；we\_s、we\_h都无效时，不进行写入操作。

ErrorEPC寄存器模块信号说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 信号类型 | 数据类型 | 位宽 | 含义 |
| clk | 输入 | wire | 1-bit | 时钟信号 |
| rst | 输入 | wire | 1-bit | 寄存器复位信号 |
| r\_p | 输入 | wire | 1-bit | 流水线读信号 |
| r\_h | 输入 | wire | 1-bit | 硬件读信号 |
| cache\_error | 输入 | wire | 1-bit | cache\_error信号 |
| we\_s | 输入 | wire | 1-bit | 软件写信号 |
| bd\_p | 输入 | wire | 1-bit | 流水线BD位 |
| write\_data | 输入 | wire | 32-bit | 写数据 |
| pc\_p | 输入 | wire | 32-bit | 流水线中pc |
| read\_data | 输出 | wire | 32-bit | 读出的数据 |
| we | 内部信号 | wire | 1-bit | 写信号 |
| re | 内部信号 | wire | 1-bit | 读信号 |
| err\_epc\_t | 内部信号 | wire | 32-bit | err\_epc中间结果 |
| err\_epc\_data | 内部信号 | wire | 32-bit | err\_epc寄存器数据 |
| err\_epc | 内部信号 | wire | 32-bit | err\_epc寄存器 |

1. BadVaddr寄存器模块说明：通过调用带使能端的寄存器来实现BadVaddr的置位。



（1）当地址错误有效时，硬件写信号有效，将发生地址错误的地址在时钟上升沿时写入BadVaddr寄存器。

（2）当流水线读信号和硬件读信号有一个有效时读出寄存器内容。

BadVaddr寄存器模块信号说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 信号类型 | 数据类型 | 位宽 | 含义 |
| clk | 输入 | wire | 1-bit | 时钟信号 |
| rst | 输入 | wire | 1-bit | 寄存器复位信号 |
| r\_p | 输入 | wire | 1-bit | 流水线读信号 |
| r\_h | 输入 | wire | 1-bit | 硬件读信号 |
| addr\_err | 输入 | wire | 1-bit | 地址错误信号 |
| badvaddr\_p | 输入 | wire | 32-bit | 流水线错误地址 |
| read\_data | 输出 | wire | 32-bit | 读出的数据 |
| we\_h | 内部信号 | wire | 1-bit | 硬件写信号 |
| re | 内部信号 | wire | 1-bit | 读信号 |
| badvaddr | 内部信号 | wire | 32-bit | BadVaddr寄存器 |

1. BadVinstr寄存器模块说明：通过调用带使能端的寄存器来实现BadVinstr的置位。



1. 当发生异常或中断时，硬件写信号有效，将发生异常或中断时将发生异常的指令在时钟上升沿时写入BadVinstr寄存器。
2. 当流水线读信号和硬件读信号有一个有效时读出寄存器内容。

BadVinstr寄存器模块信号说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 信号类型 | 数据类型 | 位宽 | 含义 |
| clk | 输入 | wire | 1-bit | 时钟信号 |
| rst | 输入 | wire | 1-bit | 寄存器复位信号 |
| r\_p | 输入 | wire | 1-bit | 流水线读信号 |
| r\_h | 输入 | wire | 1-bit | 硬件读信号 |
| exception\_abort | 输入 | wire | 1-bit | 异常信号 |
| irq | 输入 | wire | 1-bit | 中断信号 |
| badvinstr\_p | 输入 | wire | 32-bit | 流水线错误指令 |
| read\_data | 输出 | wire | 32-bit | 读出的数据 |
| we\_h | 内部信号 | wire | 1-bit | 硬件写信号 |
| re | 内部信号 | wire | 1-bit | 读信号 |
| badvinstr | 内部信号 | wire | 32-bit | BadVinstr寄存器 |

1. NestedCause寄存器通过调用Cause寄存器模块实现.

8、NestedStatus寄存器通过调用Status寄存器模块实现。

9、NestedEPC寄存器通过调用EPC寄存器模块实现。

1. NestedErrorEPC寄存器通过调用ErrorEPC寄存器模块实现。
2. NestedBadVaddr寄存器通过调用BadVaddr寄存器模块实现。
3. NestedBadVinstr寄存器通过调用BadVinstr寄存器模块实现。

13、Compare寄存器模块说明：通过调用带使能端的寄存器来实现BadVaddr的置位。

Compare寄存器中的值可以通过硬件置位，也可以通过软件置位。



Compare寄存器模块信号说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 信号类型 | 数据类型 | 位宽 | 含义 |
| clk | 输入 | wire | 1-bit | 时钟信号 |
| rst | 输入 | wire | 1-bit | 寄存器复位信号 |
| r\_p | 输入 | wire | 1-bit | 流水线读信号 |
| r\_h | 输入 | wire | 1-bit | 硬件读信号 |
| we\_h | 输入 | wire | 1-bit | 硬件写信号 |
| we\_s | 输入 | wire | 1-bit | 软件写信号 |
| write\_data | 输入 | wire | 32-bit | 写数据 |
| read\_data | 输出 | wire | 32-bit | 读出的数据 |
| we | 内部信号 | wire | 1-bit | 写信号 |
| re | 内部信号 | wire | 1-bit | 读信号 |
| compare | 内部信号 | wire | 32-bit | compare寄存器 |

1. Count寄存器模块说明：通过调用带使能端的寄存器来实现Count的置位。



Count寄存器中的计数器按照一定的频率加1，本次设计中每周期上升沿时计数器加1。当Count寄存器内容与Compare寄存器中的值相同时发生时钟中断。

Count寄存器模块信号说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 信号类型 | 数据类型 | 位宽 | 含义 |
| clk | 输入 | wire | 1-bit | 时钟信号 |
| rst | 输入 | wire | 1-bit | 寄存器复位信号 |
| r\_p | 输入 | wire | 1-bit | 流水线读信号 |
| r\_h | 输入 | wire | 1-bit | 硬件读信号 |
| we\_h | 输入 | wire | 1-bit | 硬件写信号 |
| read\_data | 输出 | wire | 32-bit | 读出的数据 |
| re | 内部信号 | wire | 1-bit | 读信号 |
| count | 内部信号 | wire | 32-bit | count寄存器 |

1. 系统栈指针寄存器模块说明：通过调用带使能端的寄存器来实现ssp的置位。

系统栈指针的设计读和写只能通过软件完成。



系统栈指针寄存器模块信号说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 信号名 | 信号类型 | 数据类型 | 位宽 | 含义 |
| clk | 输入 | wire | 1-bit | 时钟信号 |
| rst | 输入 | wire | 1-bit | 寄存器复位信号 |
| r\_p | 输入 | wire | 1-bit | 流水线读信号 |
| we\_s | 输入 | wire | 1-bit | 硬件写信号 |
| write\_data | 输入 | wire | 32-bit | 写数据 |
| read\_data | 输出 | wire | 32-bit | 读出的数据 |
| ssp | 内部信号 | wire | 32-bit | ssp寄存器 |

16、Config寄存器模块说明：通过调用带使能端的寄存器来实现Config的置位。

配置寄存器是只读寄存器，通过读配置寄存器可以查看相关配置。