**文档版本： Ver1.0**

**最后修改日期： 2016-11-7**

**修改人： zhaozhenzhen**



**E课网 – UVM实战培训**



**Lab01实验–plan**

**目 录**

1. **File information.................................................................3**

**1.1 Block information...............................................................3**

**1.2 Revision.......................................................................................3**

1. **Design dependencies.....................................................................4**

**2.1 Basic component..................................................................4**

**2.2 Top design framework........................................................4**

**2.3 Define ports..................................................................................5**

**2.4 Exe\_preprocessor functional description ............................6**

**2.5 ALU design framework.............................................................7**

**2.6 ALU functional description .....................................................8**

**3. The functional point to be verified........................................10**

**4.TB Environment.............................................................................14**

**5.Verification Requirement.....................................................14**

**6.Testcase planned.................................................................15**

**7. Coverage planned..............................................................17**

1. **File information**

**1.1 Block information**

|  |  |  |  |
| --- | --- | --- | --- |
| **Project\_name** | **Module\_name** | **Author\_name** | **Date** |
| **~ ~ ~** | **Exe\_pre and ALU** | **Zhenzhen Zhao** | **03/12/2016** |

**1.2 Revision**

|  |  |
| --- | --- |
| **Modify content** | **xxx** |
| **Modify\_date** | **06/12/2016** |
| **Modify\_author** | **Xiaoming Zhang** |

**2. Design dependencies**

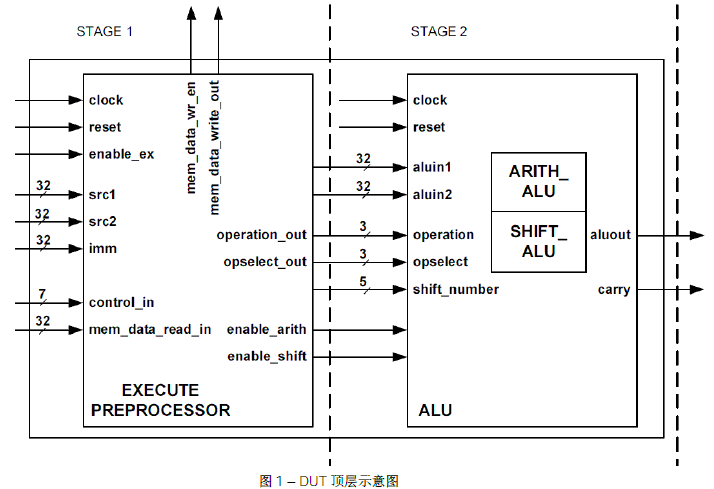
**2.1 Basic component**

**预处理器模块(Execute Preprocessor)**：对外部输入的数据和控制信号进行处理，并产生对应的ALU可接受的数据和控制信号。

**算术逻辑运算单元（ALU）**:根据输入的opselect、operation和shift\_number信号执行相关算数逻辑运算。

预处理器和ALU具有相同的时钟和复位信号。

**2.2 Top design framework**



该实验设计框架由两级流水的系统模块组成，第一级为一个“预处理器模块”，第二级为一个“算术逻辑运算单元ALU模块”。

»试验过程中，两个模块的时钟由**clock**和复位信号由**reset**来完成。

»**control\_in**为指令信号，用于控制预处理器的信号到ALU的操作。

»可实现的操作：

**Shift操作**：逻辑左移、算数左移、逻辑右移、算数右移。

**运算功能**：ADD、HADD、SUB、NOT、AND、OR、XOR、LHG.

**读操作**：LOADBYTE、LOADBYTEU、LOADHALF、LOADHALFU、LOADWORD、OTHERS.

**写操作**：mem\_data\_write\_in、mem\_write\_en.

**2.3 Define ports**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **信号名称** | **位宽** | **输入/输出** | **时钟域** | **信号描述** |
| **clock** | 1 | input | clock | 时钟信号 |
| **reset** | 1 | input | clock | 复位信号，高电平有效 |
| **Enable\_ex** | 1 | input | clock | 使能信号，高电平有效 |
| **Src1** | Register\_width | input | clock | 输入数据信号1 |
| **Src2** | Register\_width | input | clock | 输入数据信号2 |
| **imm** | Register\_width | input | clock | 多功能数据&控制信号 |
| **Control\_in** | 7 | input | clock | 指令信号 |
| **Mem\_data\_read\_in** | Register\_width | input | clock | 由memory输出的数据信号 |
| **Mem\_data\_**  **write\_out** | Register\_width | output | clock | 输出到memory的数据信号 |
| **Mem\_wirte\_en** | 1 | output | clock | 写memory使能信号，低电平有效 |
| **aluout** | Register\_width | output | clock | ALU计算结果输出数据信号 |
| **carry** | 1 | output | clock | ALU计算结果进位信号 |

**2.4 Exe\_preprocessor functional description**

**前提**：对于所有的时序逻辑输出，只有在“**enable\_ex=1**”的条件下才会发生改变。

**» aluin1**:

|  |  |
| --- | --- |
| **enable\_ex** | **aluin1** |
| **0** | **保持不变** |
| **1** | **src1** |

**»aluin2:**

|  |  |  |  |
| --- | --- | --- | --- |
| **enable\_ex** | **opselect** | **control\_in** | **aluin2** |
| **0** | **X** | **X** | **保持不变** |
| **1** | **arith\_logic** | **0** | **src2** |
| **1** | **arith\_logic** | **1** | **imm** |
| **1** | **mem\_read** | **0** | **保持不变** |
| **1** | **mem\_read** | **1** | **mem\_data\_read\_in** |
| **对于其他的信号组合，aluin2均保持不变** | | | |

**»operation\_out:**

|  |  |
| --- | --- |
| **enable\_ex** | **operation\_out** |
| **0** | **保持不变** |
| **1** | **control\_in[6:4]** |

**»opselect\_out:**

|  |  |
| --- | --- |
| **enable\_ex** | **opselect\_out** |
| **0** | **保持不变** |
| **1** | **control\_in[2:0]** |

**»shift\_number:**

|  |  |  |  |
| --- | --- | --- | --- |
| **enable\_ex** | **opselect** | **imm[2]** | **shift\_number** |
| **0** | **X** | **X** | **保持不变** |
| **1** | **shift\_reg** | **0** | **imm[10:6]** |
| **1** | **shift\_reg** | **1** | **src[4:0]** |
| **对其他的信号组合，shift\_number=0** | | | |

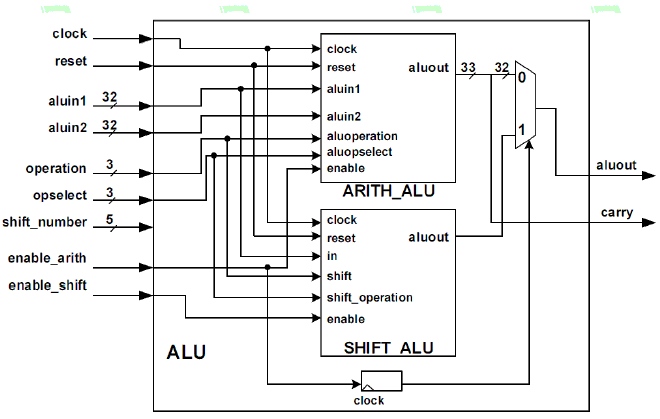
**»enable\_arith:**

|  |  |  |  |
| --- | --- | --- | --- |
| **enable\_ex** | **opselect** | **control\_in[3]** | **enable\_arith** |
| **0** | **X** | **X** | **0** |
| **1** | **arith\_logic** | **0** | **1** |
| **1** | **arith\_logic** | **1** | **1** |
| **1** | **mem\_read** | **0** | **0** |
| **1** | **mem\_read** | **1** | **1** |
| **对其他的信号组合,enable\_arith=0** | | | |

**»enable\_shift:**

|  |  |  |
| --- | --- | --- |
| **enable\_ex** | **opselect** | **enable\_shift** |
| **0** | **X** | **0** |
| **1** | **shift\_reg** | **1** |
| **对其他的信号组合，enable\_number=0** | | |

**2.5 ALU design framework**



**2.6 ALU functional description**

**前提:**所有指令信号在**enable\_arith==1**或者**enable\_shift==1**的任一条件成立下有效。

ALU主要实现算术逻辑运算（ARITH\_ALU）和移位操作(SHIFT\_ALU)的功能。

**»ARITH\_ALU:**

|  |  |  |  |
| --- | --- | --- | --- |
| **enable\_arith** | **aluopseiect** | **operation** | **aluout** |
| **0** | **X** | **X** | **保持不变** |
| **1** | **arith\_logic** | **ADD** | **带符号的加法**  **aluin1+aluin2** |
| **1** | **arith\_logic** | **HADD(16bits)** | **{h\_carry,h\_add[15:0]}=aluin1[15:0] +aluin2[15:0]**  **carry=h\_carry;**  **aluout=sxt(h\_add);** |
| **1** | **arith\_logic** | **SUB** | **带符号的减法**  **aluin1 -aluin2；** |
| **1** | **arith\_logic** | **NOT** | **对aluin2按位取反；**  **carry=0;** |
| **1** | **arith\_logic** | **AND** | **对aluin1和aluin2按位与；**  **carry=0;** |
| **1** | **arith\_logic** | **OR** | **对aluin1和aluin2按位或；**  **carry=0;** |
| **1** | **arith\_logic** | **XOR** | **对aluin1和aluin2按位异或；**  **carry=0;** |
| **1** | **arith\_logic** | **LHG** | **aluout[31:16]=aluin2[15:0];**  **aluout[15:0]=16’h0;carry=0;** |
| **1** | **mem\_read** | **LOADBYTE** | **signext➀{aluin2[7:0]};**  **carry=0;** |
| **1** | **mem\_read** | **LOADBYTEU** | **zeropad➁{aluin2[7:0]};**  **carry=0;** |
| **1** | **mem\_read** | **LOADHALF** | **signext{aluin2[15:0]};**  **carry=0;** |
| **1** | **mem\_read** | **LOADHALFU** | **zeropad{aluin2[15:0]};**  **carry=0;** |
| **1** | **mem\_read** | **LOADWORD** | **aluin2;carry=0;** |
| **1** | **mem\_read** | **OTHERS** | **aluin2;carry=0;** |
| **对于其他的信号组合,aluout和carry保持不变** | | | |

**»SHIFT\_ALU:**

|  |  |  |
| --- | --- | --- |
| **enable\_shift** | **shift\_operation[1:0]** | **aluout** |
| **0** | **X** | **保持不变** |
| **1** | **逻辑左移** | **in<<shift**  **最低位补0；**  **carry=0;** |
| **1** | **算术左移** | **in<<shift**  **最低位补0；**  **若in为正数，carry=0;**  **若in为负数，carry=1;** |
| **1** | **逻辑右移** | **in>>hift**  **最低位补0；**  **carry=0;** |
| **1** | **算术右移** | **in>>shift**  **最高位为符号位；**  **carry=0;** |
| **对于其他的信号组合，aluout保持不变** | | |

**3. The functional point to be verified**

**»**复位时，时序逻辑输出aluin1、aluin2、operation\_out、opselect\_out、shift\_number、enable\_arith、enable\_shift、aluout、carry是否保持不变。

**»**当control\_in和mem\_write\_en(低电平有效)控制memory进行写操作时，ALU是否被跳过，此时写入的数据是否为src2的输入值。

**»**对aluin1端口进行功能测试，当enable\_ex=1时，aluin1输出是否为src1.

**»**对aluin2端口进行功能测试，当enable\_ex=1时，opselect选择arith\_logic功能，控制端口control\_in输出为0/1时，aluin2输出是否为src2/imm；opselect选择mem\_read功能，控制端口control\_in 输出为0/1时，aluin2输出是否保持不变/mem\_data\_read\_in.

当控制信号control\_in输出为其他值时，是否会报错还是保持上一时刻的信号不变。

**»**对operation\_out和opselect\_out端口进行功能测试，当enable\_ex=1时，operation\_out和opselect\_out的输出分别位于control\_in[6:0]中的位置是否满足、operation\_out=control\_in[6:4],opselect\_out=control\_in[2:0]

**»**对shift\_number端口，当opselect选择shift\_reg，imm的输入为0/1时，shift\_number的输出是否为imm[10:6]/src2[4:0]；当imm输入为未知X或高阻Z状态时，shift\_number的输出又将是什么，还是保持不变。

**»**对enable\_arith端口，当enable\_ex=1,opselect选择arith\_logic时,当control\_in=0/1，enable\_arith=1；当control\_in=X/Z, enable\_arith=0；Opselect选择mem\_read时，control\_in=0/1, enable\_arith=0/1；control\_in=X/Z, enable\_arith=0。

**»**对enable\_shift端口，enable\_ex=1,opselect选择shift\_reg时, enable\_shift是否输出为1；而当enable\_ex=0,opselect输入未知，enable\_shift是否输出为0.

**»**对shift\_ALU模块进行功能点测试：enable\_shift=1时，ALU开启移位操作。

当shift\_operation[1:0]选择SHLEFTLOG,aluout是否会实现逻辑左移功能，移位后在最低位进行补零操作，此时carry输出是否为0；

当shift\_operation[1:0]选择SHLEFTART,aluout是否会实现算术左移功能，移位后在最低位进行补零操作，此时若in=正数，carry输出为何值，in=负数，carry输出又为何值；

当shift\_operation[1:0]选择SHRGHTLOG,aluout是否能实现逻辑右移功能，移位后最高位补0，此时carry输出是否为0；

当shift\_operation[1:0]选择SHRGHART,aluout是否能实现算术右移功能,且最高位为符号位,查看此时的carry的输出是否为0.

**»**对arith\_ALU模块进行功能测试:enable\_arith=1时，ALU开启运算功能操作。

当aluopselect选择arith\_logic，operation选择ADD功能，查看aluout的最终输出是否实现了带符号的加法“aluout1+aluout2”.

当aluopselect选择arith\_logic，operation选择HADD（低16位的加法）功能，查看aluout的最终输出是否实现了“h\_add[15:0]=aluin1[15:0]+aluin2[15:0]”,此时，carry=h\_carry=h\_add

当aluopselect选择arith\_logic，operation选择SUB功能，查看aluout的最终输出是否实现了带符号减法“aluin1-aluin2”.

当aluopselect选择arith\_logic，operation选择NOT功能，查看aluout的最终输出是否实现aluin2按位取反，此时的carry输出又为何值。

当aluopselect选择arith\_logic，operation选择OR功能，查看看aluout的最终输出是否实现aluin1和aluin2按位与的功能，此时carry的输出又为何值。

当aluopselect选择arith\_logic，operation选择XOR功能，查看看aluout的最终输出是否实现aluin1和aluin2按位异与的功能，此时carry的输出又为何值。

当aluopselect选择arith\_logic，operation选择LHG功能，查看看aluout的最终输出是否实现了将aluin2的值赋给aluout的高16位，aluout的低16位补零的操作，最后carry的输出是否为0。

当aluopselect选择mem\_read读操作时，operation选择LOADBYTE，查看aluout的输出是否实现了将aluin2的低8位读入signext物理地址中，此时carry的输出是否为0.

当aluopselect选择mem\_read读操作时，operation选择LOADBYTEU，查看aluout的输出是否实现了将aluin2的低8位读入zeropad物理地址中，此时carry输出是否为0.

当aluopselect选择mem\_read读操作时，operation选择LOADHALF，查看aluout的输出是否实现了将aluin2的低16位数读入signext物理地址中，此时carry输出是否为0.

当aluopselect选择mem\_read读操作时，operation选择LOADHALFU，查看aluout的输出是否实现了将aluin2的低16位数读入zeropad物理地址中，此时carry输出是否为0.

当aluopselect选择mem\_read读操作时，operation选择LOADWORD，查看aluout的输出是否实现了将aluin2的数据此时carry输出是否为0.

当aluopselect选择mem\_read读操作时，operation选择Others，查看aluout输出是否实现了将aluin2的数据此时carry输出是否为0.

**4.TB Environment**

****

**5. Verification Requirement**

Models/BFMs used

设计主要在Linux系统下用到QuestaSim软件进行操作。运用System Verilog语言进行和UVM方法学进行编程。

**6. Testcase planned**

**对于Ex\_proc模块：**

|  |  |  |  |
| --- | --- | --- | --- |
| **reset=1 reset高电平有效** | | aluin1 | 都为0 |
| aluin2 |
| operation\_out |
| opselect\_out |
| shift\_number |
| enable\_arith |
| enable\_shift |
| **!reset & !enable\_ex** | | aluin1 | 保持不变 |
| aluin2 |
| operation\_out |
| opselect\_out |
| shift\_number |
| enable\_arith | 都为0 |
| enable\_shift |
| **!reset & enable\_ex** | **opselect**分别为： ARITH\_LOGIC， MEM\_READ, SHIFT\_REG 这几种情况； **imm[2]**分别为：0,1； **control\_in[3]**分别为：0,1 | aluin1 | src1 |
| operation\_out | control\_in[6:4] |
| opselect\_out | control\_in[2:0] |
| aluin2 | 依赖opselect和 control\_in[3] |
| enable\_arith |
| shift\_number | 依赖opselect和imm[2] |
| enable\_shift | 依赖opselect |

**对于Shift\_ALU模块：**

|  |  |  |  |
| --- | --- | --- | --- |
| **reset=1 reset高电平有效** | | carry | 0 |
| aluout |
| **!reset & !enable\_shift** | | aluout | 保持不变 |
| **!reset & enable\_shift** | **shift\_operation[1:0]** 分别为： SHLEFTLOG SHRGHTLOG SHLEFTART SHRGHTLOG | aluout | 依赖于 shift\_operation [1:0] |

**对于Arith\_ALU:**

|  |  |  |  |
| --- | --- | --- | --- |
| **reset=1 reset高电平有效** | | carry | 0 |
| aluout |
| **!reset & !enable\_arith** | | aluout | 保持不变 |
| **!reset & enable\_arith** | **aluopselect**分别为： ARITH\_LOGIC MEM\_READ  **aluoperation**分别为： ADD HADD SUB NOT AND OR XOR LHG LOADBYTE LOADBYTEU LOADHALF LOADHALFU LOADWORD Others | aluout | 检查aluout的情况 |

**7. Coverage planned**

验证每个功能点是否被覆盖，且覆盖率为多少，当覆盖率达不到100%时，分析覆盖率达不到100%的原因，并建立testcase对其进行验证，知道最终测试点的功能测试覆盖率达到100%为止。