### ****16 bit MIPS CPU****

Name: YUAN Yaodong Student ID: 44231037

1. **Original Operation Added in My Design:**

*`define* MUL  5'b10010  ***// r1 <- r2 \* r3             ---R type***

*`define* DIV  5'b10110  ***// r1 <- r2/r3               ---R type***

*`define* LANDI 5'b10111  ***// r1 <- r2 and {12’b111111111111,val3}  ---RI type***

*`define* LORI  5'b10011  ***// r1 <- r2 or  {12'b000000000000,val3}  ---RI type***

*`define* LXORI 5'b10101  ***// r1 <- r2 xor {12'b000000000000,val3}  ---RI type***

* 1. **MUL is a multiplication operation for 2 regs：r2, r3 ( R type)**

And in ALU, it may compute by reg\_A and reg\_B.

The value of Reg\_A and reg\_B are decided by the type of instruction.

        `MUL    :

            {cf\_next, ALUo} *=* {1'b0, reg\_A} *\** {1'b0, reg\_B};

图形用户界面

中度可信度描述已自动生成

Which is the writing logic for **MUL, LANDI, LORI, LXORI, DIV.**

**Imem\_MUL.ini :**

文本

描述已自动生成

**Dmem\_MUL.mem:**

图形用户界面, 文本

描述已自动生成

* 1. **Likelihood, DIV is a dividend operation ( R type instrutction)**

        `DIV    :

            {cf\_next, ALUo} *=* {1'b0, reg\_A} */* {1'b0, reg\_B};

**Imem\_DIV.ini :**

电脑的屏幕截图

描述已自动生成

The Data in Memory is the same as the **MUL.**

**Dmem\_DIV.mem:**

图形用户界面, 文本

描述已自动生成

Next 3 operations are logic operations, and all of them are RI type instructions.

* 1. **LANDI: r1 <- r2 and {12'b111111111111,val3}**

*else* *begin*

*if*((id\_ir[15:11] *==* `LOAD) *||* (id\_ir[15:11] *==* `SLL) *||* (id\_ir[15:11] *==* `SRL) *||* (id\_ir[15:11] *==* `SLA) *||* (id\_ir[15:11] *==* `SRA) *||* (id\_ir[15:11] *==* `LORI) *||* (id\_ir[15:11] *==* `LXORI))

***// non-writing instructions***

                 reg\_B *<=* {12'b000000000000, id\_ir[3:0]};

*else* *if*((id\_ir[15:11] *==* `LANDI))

                reg\_B *<=* {12'b111111111111, id\_ir[3:0]}; ***//reg\_B <= 12'b1,val3***

**because we set the Upper 12 bit are all 1, so reg\_B should change the value based on the instruction, if it is LANDI**

        `LANDI    : *begin*

            ALUo *=* reg\_A *&* {12'b111111111111,reg\_B};

            cf\_next *=* 1'b0;

*end*

**Imem\_LANDI.ini :**

文本

描述已自动生成

**Dmem\_LANDI.ini :**

图形用户界面, 文本

描述已自动生成

**Dmem\_LANDI.mem:**

文本

描述已自动生成

* 1. **LORI: r1 <- r2 OR {12'b000000000000,val3}**

**The Data in Memory are the same as before.**

        `LORI    :  *begin*

            ALUo *=* reg\_A *|* {12'b000000000000, reg\_B};

            cf\_next *=* 1'b0;

*end*

**Imem\_LORI.ini :**

文本

描述已自动生成

**Dmem\_LORI.mem:**

文本

描述已自动生成

* 1. **LXORI: r1 <- r2 XOR {12'b000000000000,val3}**

        `LXORI    :  *begin*

            ALUo *=* reg\_A *^* {12'b000000000000, reg\_B};

            cf\_next *=* 1'b0;

*end*

**Imem\_LXORI.ini :**

文本

描述已自动生成

**Dmem\_LXORI.mem:**

图形用户界面, 文本

描述已自动生成

**Time constraints:**

图形用户界面, 应用程序, Word

描述已自动生成

1. **Consideration and Impression:**

The course on FPGA design taught us through the intricate process of hardware design using Verilog, and even venturing into the CPU design. One of the primary challenges I think the most complicated units are data hazards and control hazards. The critical logic judgments required for key data transfers seemed overwhelmingly complex and difficult to comprehend, as well as the branch detection.

As I reflect from my learning experience, I am considering using my spare time and the winter vacation to further explore the basics of CPU design. I plan to read some specialized books on the subject to enhance my understanding and overcome the complexities.