**Fuyao\_fpga设计报告**

学校：安徽新华学院

姓名： 张志

一、设计简介

本次设计主要完成了22条mips指令的CPU，目前实现的功能包括上板跑完大赛提供的监控程序以及大赛个人赛的三个测试任务。其实现的特色在于支持延迟槽，在处理结构相关性时采用状态机来阻塞时钟。

二、设计方案

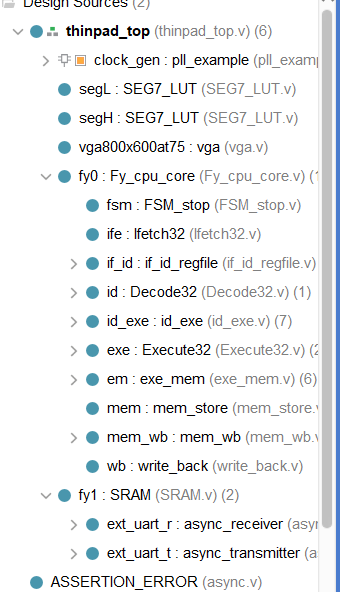
（一）总体设计思路

顶层模块thinpad\_top包括三个部分，一个是SRAM接口部分。一个是CPU部分，还有一个就是串口转发部分。而CPU部分又包括基本的五个单元，即取指令（Ifetch32），这个模块主要负责对分支地址的处理。译码（Decode32），这个模块主要负责对指令的解析和得到传递给执行单元的操作数（即处理数据相关性采用定向法）。执行（Execute32），这个模块主要是负责处理各种指令的计算。存储（mem\_store），这个模块主要是负责对于传到SRAM的数据和传入到CPU的一些地址和数据的处理，计算出传出的地址以及接受从SRAM到来的数据。回写（write\_back）。同时为了解决结构相关性，引入了状态机（FSM\_stop）。这个模块主要是通过一个状态的变迁来处理的，比如取指令在第一个时钟阶段发生，如果是访存指令那么得到第四个时钟阶段才能取消阻塞。寄存器文件（regfile32）用于读写数据。还有4个流水间寄存器。接下来就是内存部分与串口部分，这两个部分是参考了去年的比赛代码，将这两个部分结合在一起构成一个完整的模块（SRAM），这个模块在SRAM读写上主要是提供一些读写信号以及读出写入数据的处理。

三、设计结果

（一）设计交付物说明

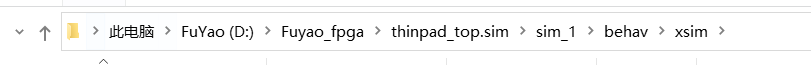
1.主目录层次位thinpad\_top，其下包括CPU部分，串口部分与SRAM接口部分。如下图所示。



2. FSM\_stop位有限状态机用于控制取指单元的地址的传递，可以用于阻塞流水线。Ifetch32是取指单元，负责给出取指的地址，而if\_id\_regfile是流水间寄存器，这个是用于传递给下一级的指令和地址，而指令来自SRAM。Decode32是译码单元，负责解决数据相关性和一些相关逻辑器件的控制信号。Id\_exe是译码单元通向执行单元的流水间寄存器。负责传递写地址，写数据，写使能，两个操作数。Execute32是执行单元，负责计算运算类指令的结果和传给访存模块的地址和写入存储器的数据。Exe\_mem是执行单元通往存储单元的一些数据和地址，比如写上存储器的数据。访存单元mem\_store是用于得到一些访存的控制信号与访存的地址。Mem\_wb是存储级通向回写级的寄存器。Wb是回写到寄存器文件里面的一些信号和数据，地址。

|  |  |
| --- | --- |
| 模块 | 作用 |
| FSM\_stop | 有限状态机用于控制取指单元的地址的传递 |
| Ifetch32 | 取指单元 |
| Decode32 | 译码单元 |
| Execute32 | 执行单元 |
| mem\_store | 访存模块 |
| Wb | 回写模块 |
| SRAM | 内存与串口模块 |
| Fuyao\_cpu\_core | CPU模块 |
| Thinpad\_top | 顶层模块 |

3． 在仿真时，先下载kernel.bin文件，然后将此文件放在如图所示的路径下。

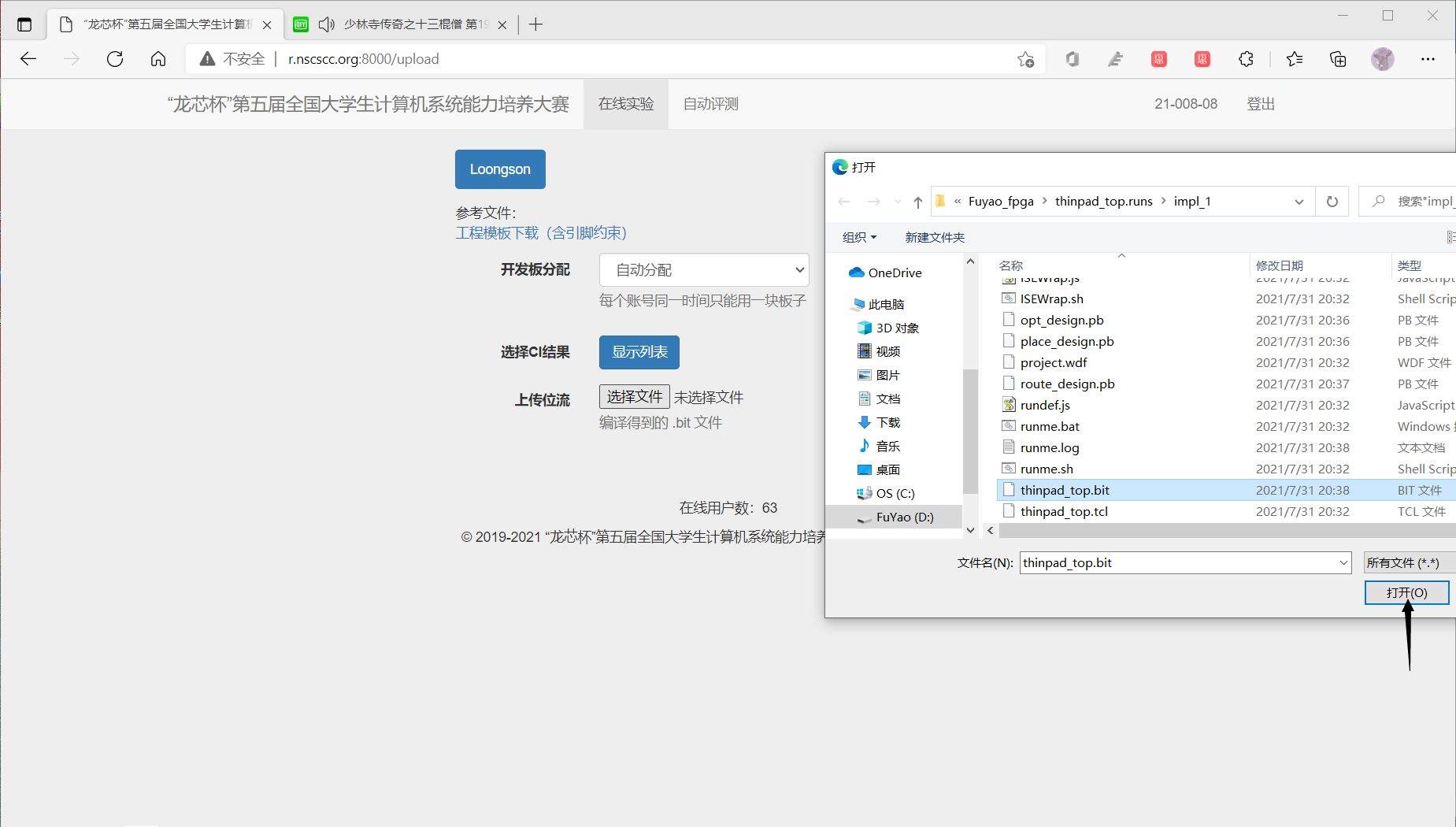


接下来就可以直接点击开始行为仿真。

在上板实验时要先生成.bit文件。记住此文件的位置，在如图所示的路径下。



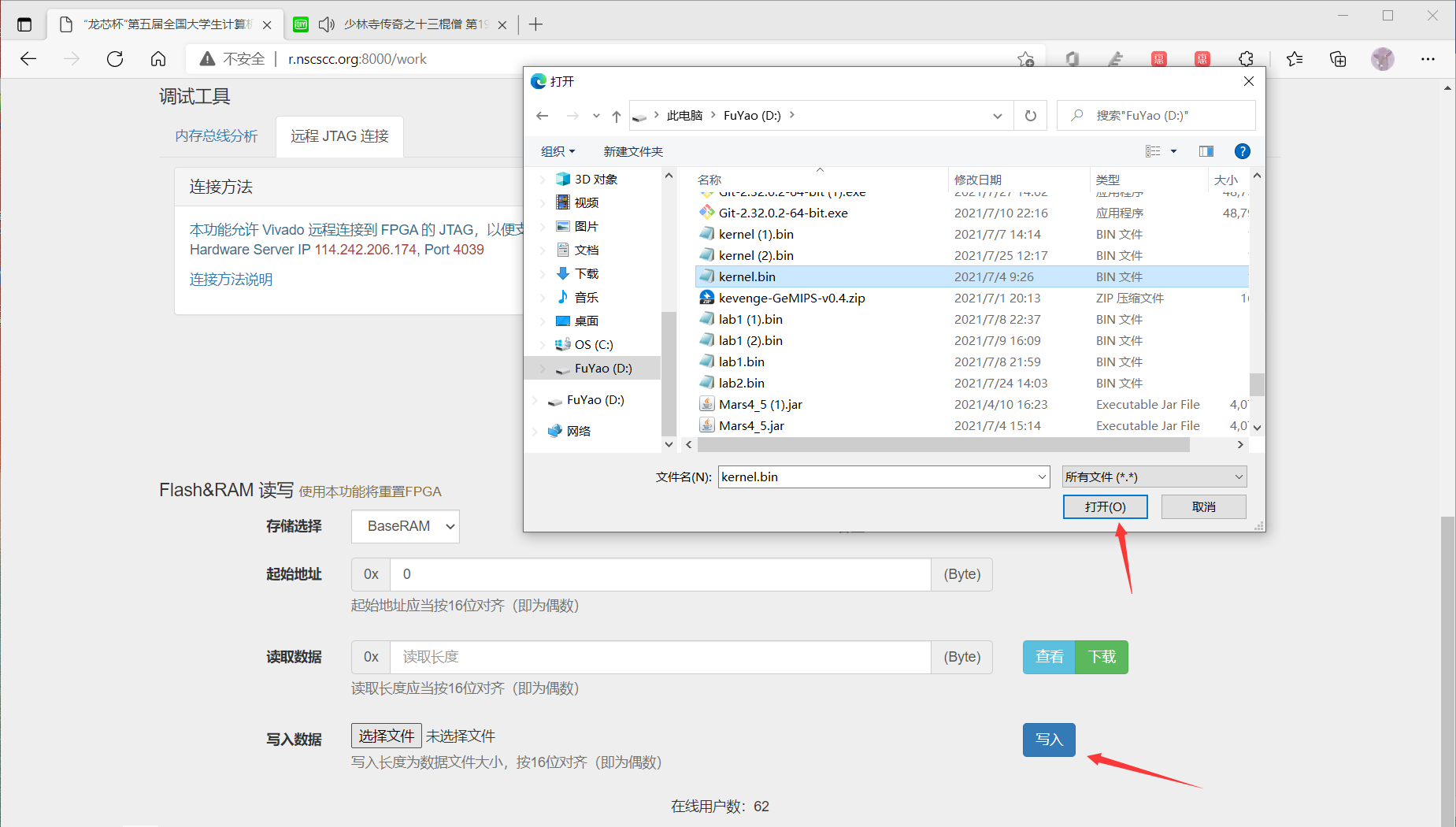
在上板操作时。先将.bit文件打开然后点击上传开始。



点击开始后可在线上开始实验。



先将kernel.bin文件写入baseram。点击写入。如果CPU设计时没有问题，那么在串口就会显示欢迎信息。

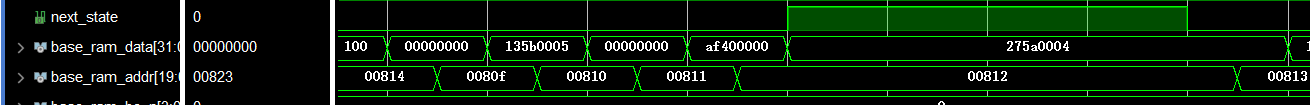


（二）设计演示结果

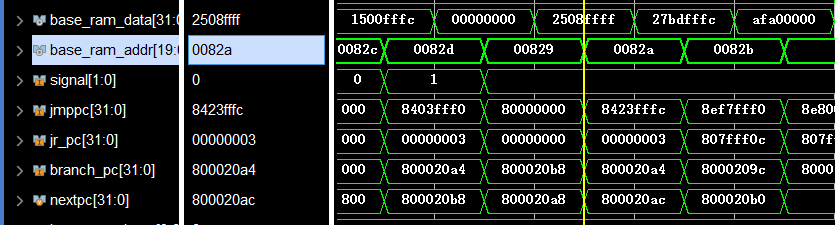
如下图所示这是在CPU上跑一个监控程序的结果，显示出欢迎字符。



下图是一个处理结构相关性的仿真图，可以看到next\_state检测到访存指令时自动变为高电平并且持续4个时钟周期。

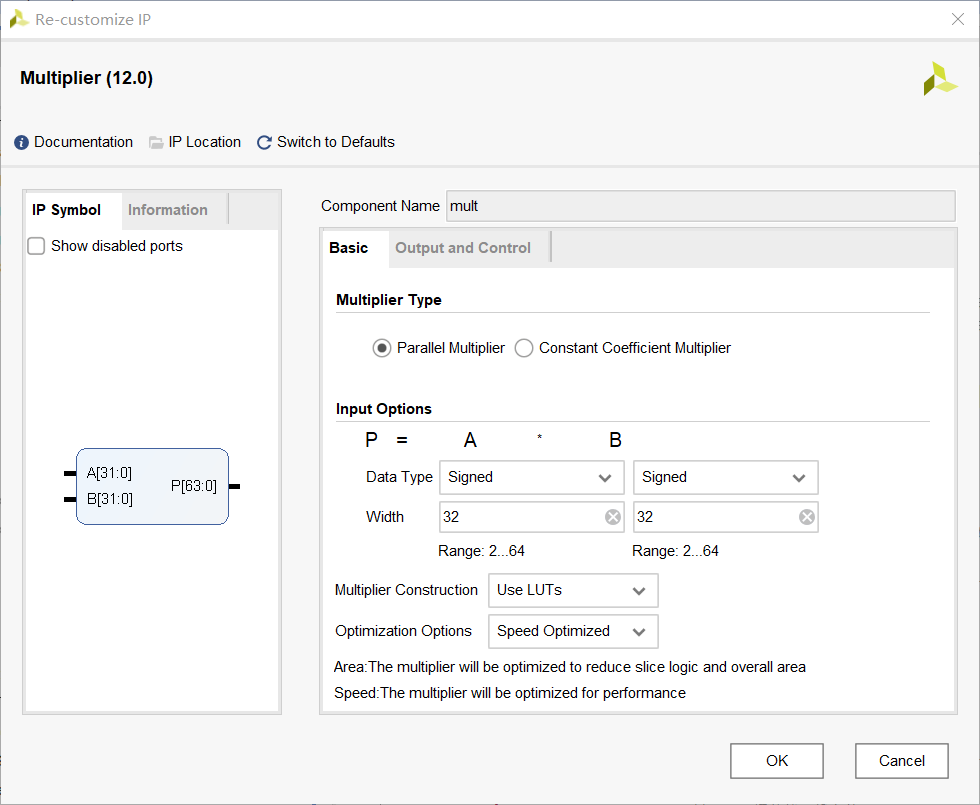


下图是一个处理延迟槽时的仿真图。可以看到当接收到跳转指令时。跳转信号signal信号变为1，这里的1表示有条件跳转指令。此时0x800020a4地址转换为0x00829.



四、参考设计说明

1.乘法部分：在设计乘法时采用的是Vivado里面自带的乘法的ip核。



2.串口部分

在设计串口通信模块参考了去年的开源代码，其网址为：

（https://gitee.com/kevenge/GeMIPS/tree/master/sources\_1/GeMIPS）

具体的信号如下：

//直连串口信号

output wire txd, //直连串口发送端

input wire rxd, //直连串口接收端

/// 处理读取或者写入的数据范围

wire is\_SerialStat = (ram2\_addr\_i == `SerialStat);

wire is\_SerialDate = (ram2\_addr\_i == `SerialDate);

wire is\_base\_ram = is\_SerialStat != 1'b1 && is\_SerialDate != 1'b1 && (ram2\_addr\_i >= 32'h80000000) && (ram2\_addr\_i < 32'h80400000);

wire is\_ext\_ram = is\_SerialStat != 1'b1 && is\_SerialDate != 1'b1 && (ram2\_addr\_i < 32'h80800000) && (ram2\_addr\_i >= 32'h80400000);

reg[31:0] serial\_o;

wire[31:0] base\_ram\_o;

wire[31:0] ext\_ram\_o;

/// 处理串口

always @(\*) begin

if(rst) begin

ext\_uart\_start <= 1'b0;

serial\_o <= 32'h0000\_0000;

ext\_uart\_tx <= 8'h00;

end

else begin

if(is\_SerialStat) begin /// 获取串口状态

serial\_o <= {{30{1'b0}}, {ext\_uart\_ready, !ext\_uart\_busy}};

ext\_uart\_start <= 1'b0;

ext\_uart\_tx <= 8'h00;

end

else if(ram2\_addr\_i == `SerialDate) begin /// 获取（或发送）串口数据

if(ram2\_we\_i) begin /// 读数据，即接收串口数据

serial\_o <= {24'h000000, ext\_uart\_rx};

ext\_uart\_start <= 1'b0;

ext\_uart\_tx <= 8'h00;

end

else begin /// 写数据，即发送串口数据

ext\_uart\_tx <= ram2\_data\_i[7:0];

ext\_uart\_start <= 1'b1;

serial\_o <= 32'h0000\_0000;

end

end

else begin

ext\_uart\_start <= 1'b0;

serial\_o <= 32'h0000\_0000;

ext\_uart\_tx <= 8'h00;

end

end

end

/// 处理串口接收的clear

reg ext\_uart\_clear\_next;

reg[3:0] ext\_uart\_clear\_para;

always @(negedge clk\_50M) begin

if(rst) begin

ext\_uart\_clear\_next <= 1'b0;

end

else begin

if(ext\_uart\_ready && ram2\_addr\_i == `SerialDate && ram2\_we\_i && ext\_uart\_clear\_next == 1'b0) begin

ext\_uart\_clear\_next <= 1'b1;

end

else if (ext\_uart\_clear == 1'b1) begin

ext\_uart\_clear\_next <= 1'b0;

end

else begin

ext\_uart\_clear\_next <= ext\_uart\_clear\_next; end end end

always @(posedge clk\_50M) begin

if(rst) begin

ext\_uart\_clear <= 1'b0;

end

else begin

if(ext\_uart\_clear\_next) begin

ext\_uart\_clear <= 1'b1;

end

else begin

ext\_uart\_clear <= 1'b0;

end

end

end

3.模块输出部分

always @(\*) begin

if(rst) begin

ram2\_data\_o <= 32'h0000\_0000;

end

else begin

if(is\_SerialStat || is\_SerialDate ) begin ram2\_data\_o <= serial\_o;

end

else if (is\_base\_ram) begin

// ram2\_data\_o <= base\_ram\_o;

case (ram2\_sel\_i)

4'b1110: begin ram2\_data\_o <= {{24{base\_ram\_o[7]}}, base\_ram\_o[7:0]};

end

4'b1101: begin ram2\_data\_o <= {{24{base\_ram\_o[15]}}, base\_ram\_o[15:8]};

end

4'b1011: begin

ram2\_data\_o <= {{24{base\_ram\_o[23]}}, base\_ram\_o[23:16]};

end

4'b0111: begin ram2\_data\_o <= {{24{base\_ram\_o[31]}}, base\_ram\_o[31:24]};

end

4'b0000: begin ram2\_data\_o <= base\_ram\_o;

end

default: begin

ram2\_data\_o <= base\_ram\_o;

end

endcase

end

else if (is\_ext\_ram) begin

case (ram2\_sel\_i)

4'b1110: begin ram2\_data\_o <= {{24{ext\_ram\_o[7]}}, ext\_ram\_o[7:0]};

end

4'b1101: begin

ram2\_data\_o <= {{24{ext\_ram\_o[15]}}, ext\_ram\_o[15:8]};

end

4'b1011: begin ram2\_data\_o <= {{24{ext\_ram\_o[23]}}, ext\_ram\_o[23:16]};

end

4'b0111: begin ram2\_data\_o <= {{24{ext\_ram\_o[31]}}, ext\_ram\_o[31:24]};

end

4'b0000: begin ram2\_data\_o <= ext\_ram\_o; end

default: beginram2\_data\_o <= ext\_ram\_o; end

endcase

end

else begin ram2\_data\_o <= 32'h0000\_0000;

end

end

end

4.信号定义部分：

在接口信号定义中各个单元的参考信号如下：执行单元：

output reg[3:0] mem\_op,

output reg[31:0] mem\_addr\_o,

output reg[31:0] mem\_data\_o

存储单元：

input wire rst,

input wire[3:0] mem\_op,

input wire[31:0] mem\_addr\_i,

input wire[31:0] mem\_data\_i,

input wire[31:0] ram\_data\_i,

output reg[31:0] mem\_addr\_o,

output reg[31:0] mem\_data\_o,

output reg mem\_we\_o,

output reg mem\_ce\_o,

output reg[3:0] mem\_sel\_o

回写单元如下：

回写 写寄存器地址，信号，数据。

input wire wd\_i,

input wire[4:0] waddr\_i,

input wire[31:0] wdata\_i,

output wire wd\_o,

output wire[4:0] waddr\_o,

output wire[31:0] wdata\_o

在实现对SRAM操作时，在访存单元参考了一下部分

always@(\*) begin

if(rst) begin

we\_o <= 1'b0; waddr\_o <= 5'b00000;

wdata\_o <= 32'h00000000;mem\_addr\_o <= 32'b00000000;

mem\_data\_o <= 32'b00000000; mem\_we\_o <= 1'b0;mem\_ce\_o <= 1'b0;

end

else begin

we\_o <= we\_i；waddr\_o <= waddr\_i;

case (mem\_op)

`MEM\_LB: begin

mem\_addr\_o <= mem\_addr\_i; mem\_data\_o <= 32'b00000000;mem\_we\_o <= 1'b1;

mem\_ce\_o <= 1'b1;

wdata\_o <= ram\_data\_i;

end

`MEM\_LW: begin

mem\_addr\_o <= mem\_addr\_i;mem\_data\_o <= 32'b00000000;

mem\_we\_o <= 1'b1; mem\_ce\_o <= 1'b1; wdata\_o <= ram\_data\_i;

end

`MEM\_SB: begin

mem\_addr\_o <= mem\_addr\_i;

mem\_data\_o <= mem\_data\_i; mem\_we\_o <= 1'b0;

mem\_ce\_o <= 1'b1;wdata\_o <= 32'b00000000;

end

`MEM\_SW: begin

mem\_addr\_o <= mem\_addr\_i; mem\_data\_o <= mem\_data\_i; mem\_we\_o <= 1'b0;

mem\_ce\_o <= 1'b1;wdata\_o <= 32'b00000000;

end

default: beginwdata\_o <= wdata\_i;

mem\_addr\_o <= 32'b00000000; mem\_data\_o <= 32'b00000000; mem\_we\_o <= 1'b0;mem\_ce\_o <= 1'b0;

end

endcase

end

end

五、参考文献

[1]杨全胜，钱瑛，任国林，王晓蔚，吴强[M]. 北京：清华大学出版社.2020.

[2]杨全胜. 计算机系统综合课程设计[M]. 北京：清华大学出版社，2008.

[3] https://gitee.com/kevenge/GeMIPS/tree/master/sources\_1/GeMIPS.