中国科学技术大学

RISC-V 五级流水线处理器设计以及拓展乘法指令 MULW

姓　名：[redacted]

学　号：[redacted]

学　院：微电子学院

专　业：电子信息

日　期：2023/07/01

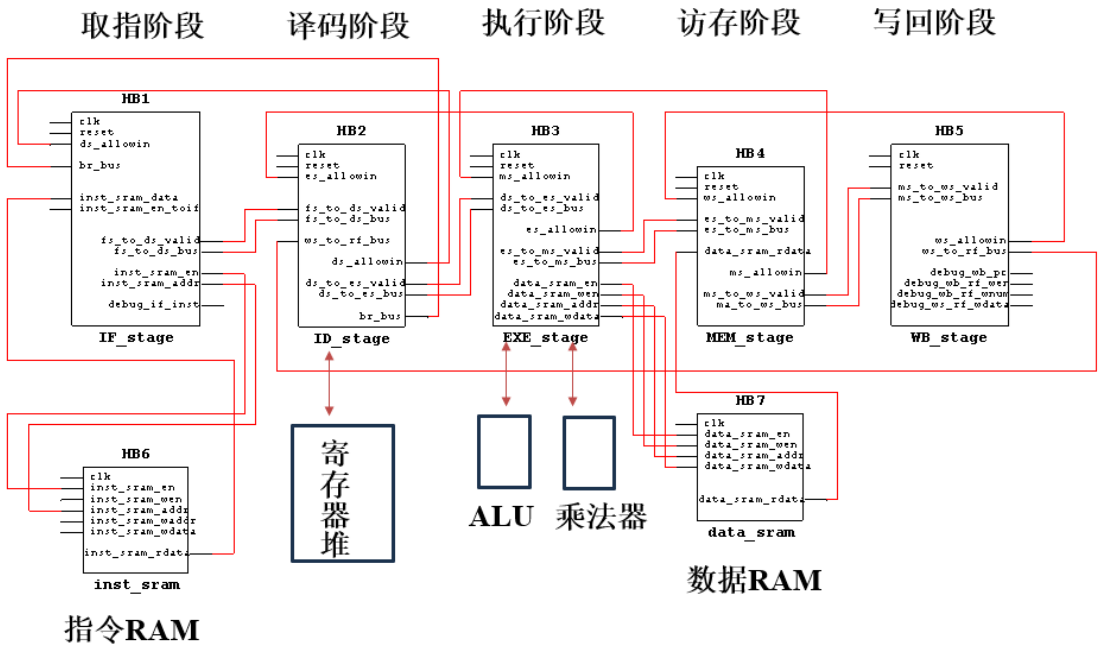
一、微架构设计

参考《计算机组成与设计-硬件/软件接口》、《CPU 设计实战》、《The RISC-V Instruction Set Manual Volume I: Unprivileged ISA》、“龙芯MIPS 处理器源码”设计了一个支持 RISC-V 部分基本指令的 64 位单发射五级流水线处理器。

目前支持的指令可以从指令译码阶段的译码逻辑中查看，如下图所示展示了 ID_stage.v 文件里面的译码逻辑部分。可以看到目前支持的指令有RV32I 基本指令集中的ADD、SUB、SLT、SLUT、AND、OR、XOR、NOR、SLL、SRL、SRA、ADDI、LUI、LD、SD、BEQ、BNE、JAL，以及 RV64M 标准拓展指令中的 MULW。

```
assign inst_add    = ( (opcode == 7'b0110011) && (func3 == 3'b000) && (func7 == 7'h0000000) )? 1 : 0;
assign inst_sub    = ( (opcode == 7'b0110011) && (func3 == 3'b000) && (func7 == 7'b0100000) )? 1 : 0;

assign inst_slt    = ( (opcode == 7'b0110011) && (func3 == 3'b010) && (func7 == 7'b0000000) )? 1 : 0;

assign inst_sltu   = ( (opcode == 7'b0110011) && (func3 == 3'b011) && (func7 == 7'b0000000) )? 1 : 0;
assign inst_and    = ( (opcode == 7'b0110011) && (func3 == 3'b111) && (func7 == 7'b0000000) )? 1 : 0;
assign inst_or     = ( (opcode == 7'b0110011) && (func3 == 3'b110) && (func7 == 7'b0000000) )? 1 : 0;
assign inst_xor    = ( (opcode == 7'b0110011) && (func3 == 3'b100) && (func7 == 7'b0000000) )? 1 : 0;
assign inst_nor    = 0;
assign inst_sll    = ( (opcode == 7'b0110011) && (func3 == 3'b001) && (func7 == 7'b0000000) )? 1 : 0;
assign inst_srl    = ( (opcode == 7'b0110011) && (func3 == 3'b101) && (func7 == 7'b0000000) )? 1 : 0;
assign inst_sra    = ( (opcode == 7'b0110011) && (func3 == 3'b101) && (func7 == 7'b0100000) )? 1 : 0;

assign inst_addi   = ( (opcode == 7'b0010011) && (func3 == 3'b000) )? 1 : 0;
assign inst_lui    = ( (opcode == 7'b0110111) )? 1 : 0;
assign inst_ld     = ( (opcode == 7'b0000011) && (func3 == 3'b011) )? 1 : 0;
assign inst_sd     = ( (opcode == 7'b0100011) && (func3 == 3'b011) )? 1 : 0;
assign inst_lw     = ( (opcode == 7'b0000011) && (func3 == 3'b010) )? 1 : 0;
assign inst_sw     = ( (opcode == 7'b0100011) && (func3 == 3'b010) )? 1 : 0;
assign inst_beq    = ( (opcode == 7'b1100011) && (func3 == 3'b000) )? 1 : 0;
assign inst_bne    = ( (opcode == 7'b1100011) && (func3 == 3'b001) )? 1 : 0;
assign inst_jal    = ( (opcode == 7'b1101111) )? 1 : 0;
assign inst_jalr   = ( (opcode == 7'b1100111) && (func3 == 3'b000))? 1 : 0;


assign inst_slti   = ( (opcode == 7'b0010011) && (func3 == 3'b010) )? 1 : 0;
assign mulw        = ( (opcode == 7'b0111011) && (func3 == 3'b000) && (func7 == 7'b0000001) )? 1 : 0;
```

所设计的处理器结构框图如下图所示：包含了五个处理阶段以及指令 RAM、数据 RAM、寄存器堆、ALU、乘法器。其中寄存器堆中有 32 个 64 位物理寄存器。



流水线的控制方式采用类似总线的通信机制，利用各级的 allowin 来向前一级发出允许输入的信号，各级的 valid 信号发送给后一级来表示当前的操作是有效的。各个子模块的端口信息可以从上面的结构框图中看到。

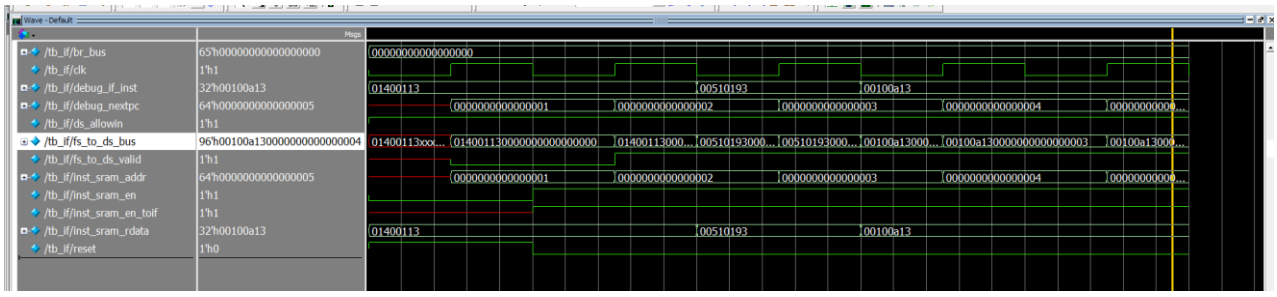
二、RTL 子模块的仿真

对每一级流水线、寄存器堆、指令 RAM、数据 RAM、ALU 进行了 RTL 仿真。

1、IF Stage RTL 建模以及仿真

RTL 仿真：顺序取指令时的波形以及相应的激励条件。

```
initial begin
    reset = 1'b1;
    ds_allowin=1'b1;
    br_bus =65'd0;
    inst_sram_rdata=32'b000000010100_00000_000_00010_0010011;
    #10
    reset =1'b0;
    inst_sram_en_toif=1'b1;
    #10
    inst_sram_rdata = 32'b000000000101_00010_000_00011_0010011;
    #10
    inst_sram_rdata = 32'b000000000001_00000_000_10100_0010011;
end
```

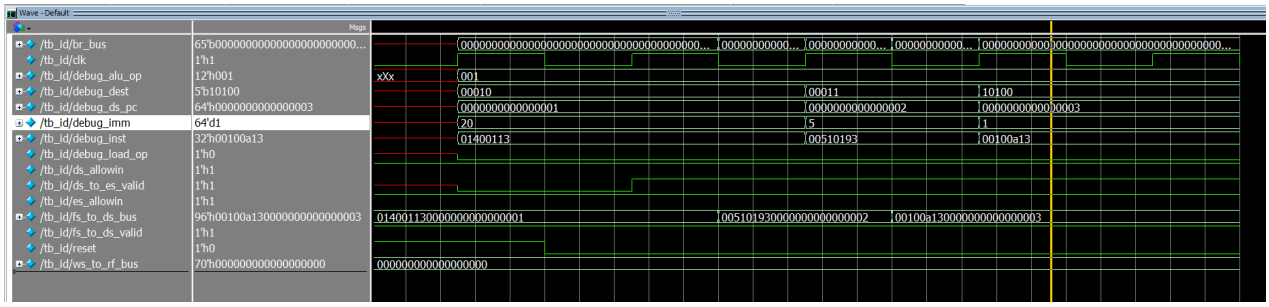


fs_to_ds_bus 是指从 IF_stage 传到 ID_stage 的总线，96 位，前 32 位表示指令，后 64 位表示当前读指令 ram 的地址。从 debug_if_inst，debug_nexpc，fs_to_ds_bus 可以看出取指阶段可以正确接收指令 ram 的读出信号，能够正确的将信息发送到输出总线端口。并且下一次取指令的地址能够根据 br_bus 正确的决定。br_bus 是分支总线（包括 1 位标志位，标志是否分支，以及跳转的 PC64 位）。

IF_stage 可以按照预期方式工作。

2、ID Stage RTL 建模以及仿真

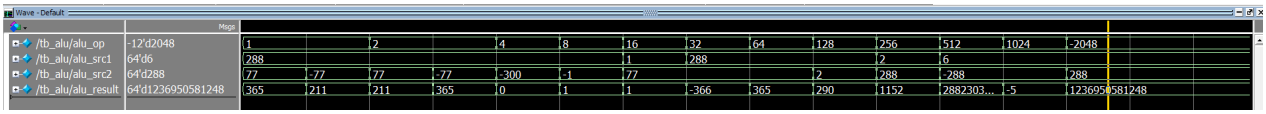
ID_stage 的主要任务是根据取指阶段传送过来的指令进行译码，产生相应的控制信息，读取寄存器堆中的数据，将数据和控制信息以总线的方式发送给第三级。这里针对了 addi 指令的取值进行了验证。后面各个阶段的仿真激励信号都通过上一级的输出总线来施加。



图中debug_alu_op 是运算类型的控制信号，debug_dest 是目标寄存器的编号，debug_ds_pc 是当前阶段指令所在的地址，debug_imm 是经过符号拓展的立即数，debug_inst 是当前的要译码的指令，ds_to_es_bus 是输出到执行阶段的控制信号和数据总线。更详细的解释放在了代码的注释里。

从上面的波形图可以看出译码阶段的逻辑正确。

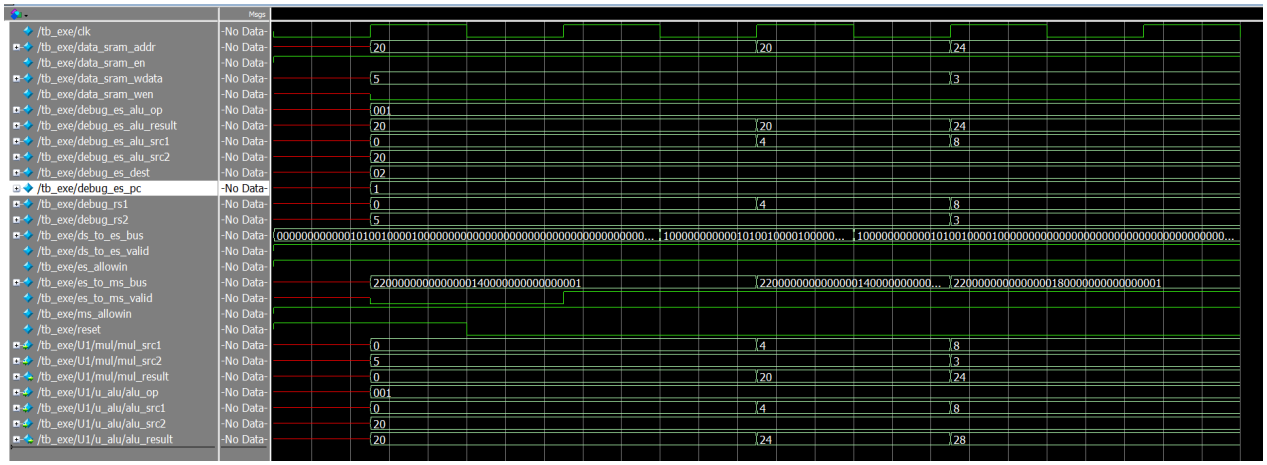
3、算术逻辑单元的 RTL 仿真



alu_op 是计算方式的控制信号，图中依此进行了加法、减法、有符号比较、无符号比较、按位与、按位或非、按位或、按位异或、逻辑左移、逻辑右移、算数右移、立即数置于高位部分的验证。可以看出结果没有问题。

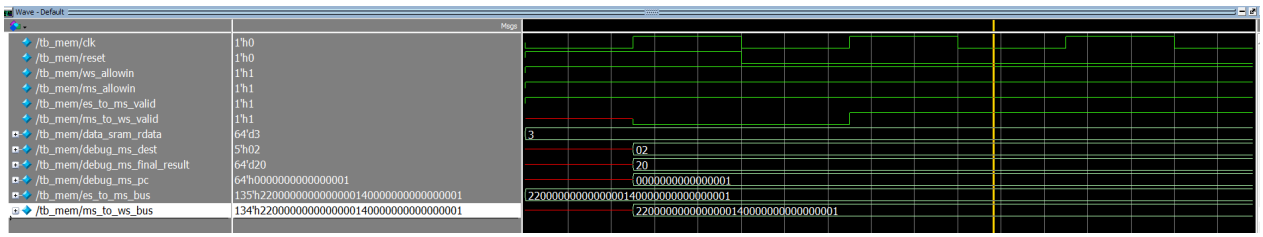
4、Exe stage 仿真

图中展示了对执行阶段的仿真模拟的波形结果。debug_es_alu_result 表示运算的结果，ds_to_es_bus 包含 exe 阶段执行所需要的信息。从给出的 testbench 文件里可以看出，预期的操作是 0+20=20，4*5=20，3*8=24。波形符合预期的结果。



5、mem stage 仿真

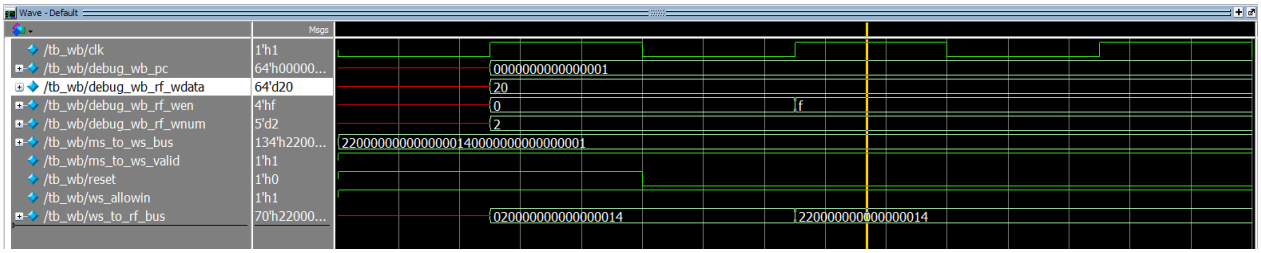
Mem 阶段主要是从存储器中读数据，根据情况确定传送给写回阶段的结果是来自存储器读出的数据还是执行阶段计算得出的数据。



显然，这个波形图表示逻辑功能没有问题。

6、wb stage 仿真

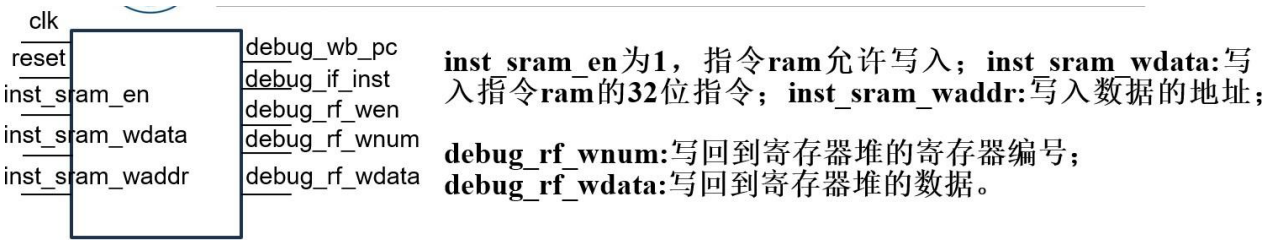
写回阶段的仿真激励信号同样通过总线输入。下面的波形同样说明了写回阶段可以正确执行。



三、CPU 的程序级仿真

在对各个流水线阶段进行仿真发现问题，并及时解决后，开始对整个 CPU 进行程序级的仿真。程序级仿真的思路是先向指令 RAM 中写入接下来需要执行的指令，然后关闭指令 RAM 的写使能，打开 CPU，使 CPU 从第一条指令地址处开始取指令工作。观察程序执行完成后最终的写回阶段的 debug 信号端口上的信号是否满足预期的结果。

下面这个图表达了我用于进行程序级仿真的 CPU 框图，以及相应的用于 debug 而预留的端口。



第一次模拟的指令如下所示：

首先花费 4 个上升沿向指令 ram 中写入 4 条指令：

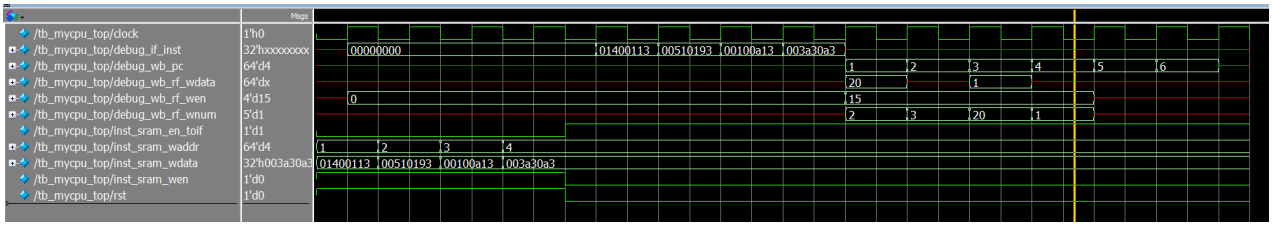
1、 addi x2 , x0 , 20

2、 addi x3 , x2 , 5

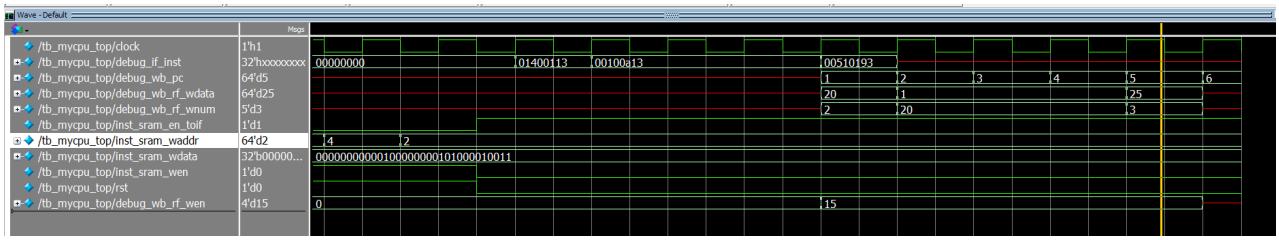
3、 addi x20, x0 , 1

4、 sd x3, 1(x20)

该模拟的波形情况如下：



从波形图上可以发现第二条指令的执行结果并没有按照预期的那样向 x3 写回 25，而是出现了未知状态。原因在于第二条指令中的数据 x2 依赖第一条指令的结果，在第二条指令



执行时，第一条指令没能将结果及时返回到 x2。目前这个处理器只能将有数据依赖关系的指令停顿 2 个周期才能正确执行。而且必须手动给汇编代码插入 nop 指令。如下图所示，可以正确运行了，更进一步的设计可以考虑实现前递逻辑，以及由硬件进行检测和停顿。

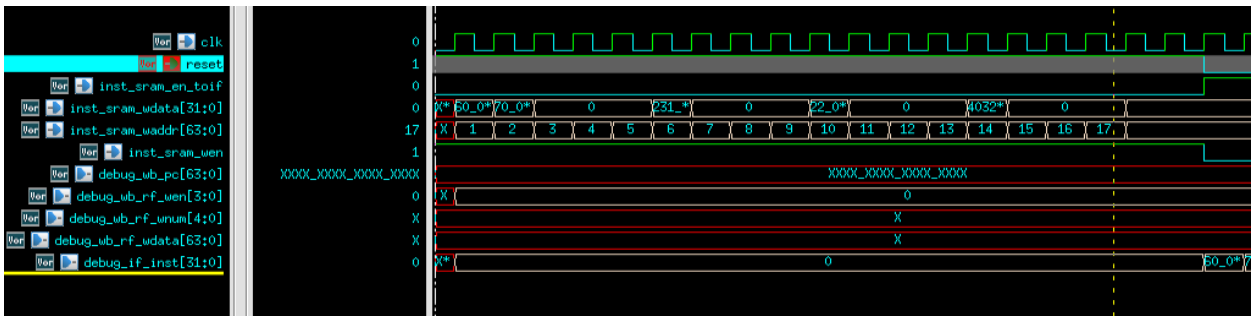
了解了数据依赖的影响后，在更进一步的仿真模拟中都在需要的地方插入了 nop 指令。在下面这个模拟中利用一个简单的代数表达式来检验 CPU 能否正确工作。

代数表达式：计算 a * b + a － b + 5，假设 a=6，b=7.

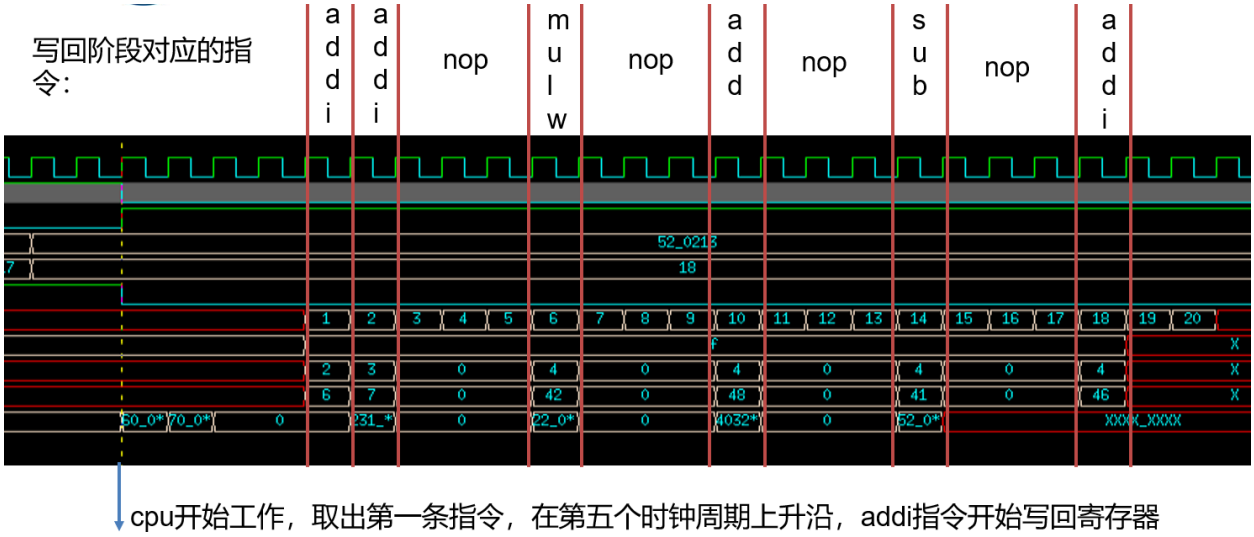
编写相应的汇编代码：

```
addi x2, x0, 6     //将 6 写入 2 号寄存器
addi x3, x0, 7     //将 7 写入 3 号寄存器
nop
nop
nop
mulw x4, x2, x3 //a*b，结果写入 4 号寄存器
nop
nop
nop
add x4, x4, x2         //从 a*b 的结果中加上 a
nop
nop
nop
sub x4, x4, x3     //从 a*b+a 中减去 b
nop
nop
nop
addi x4, x4, 5 //加上 5
```

将上面的汇编代码转成机器码后，通过多个时钟周期将指令的机器码写入到指令 RAM 中去。每个时钟周期写入一条指令，之后关闭指令 RAM 的写使能，打开 CPU，开始工作，这个过程的波形图如下图所示。



之后，开启 CPU 后，取出第一条指令，在第 5 个时钟周期时写回信号产生有效输出，当程序执行完第 18 条指令时，写回信号表示向 4 号寄存器写入值 46，与预期的结果一致。具体的波形图如下图所示。



四、DC 综合

逻辑综合使用 design compiler 软件进行。使用的工艺库是 smic018。分别在 100MHz、200MHz、250MHz 的时钟频率下进行综合。这三种情况都满足时序约束。更详细的报告放在了 result 文件夹里。

当时钟频率在 250MHz 时，资源消耗情况，预估的面积，功耗情况分别如下所示。

```
Number of ports:                          868
Number of nets:                         53044
Number of cells:                        52148
Number of combinational cells:          37158
Number of sequential cells:             14984
Number of macros/black boxes:               0
Number of buf/inv:                       4382
Number of references:                     137
```

```
Area
-----------------------------------
Combinational Area:    690005.136968
Noncombinational Area:
                       816614.578754
Buf/Inv Area:           36959.630993
Total Buffer Area:          5601.66
Total Inverter Area:       31357.97
Macro/Black Box Area:      0.000000
Net Area:                  0.000000
-----------------------------------
Cell Area:             1506619.715722
Design Area:           1506619.715722
```

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs	Cell Count
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
clock_network	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
register	240.6276	4.2971e-02	2.8109e+06	240.6734	(99.70%)		14984
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
combinational	0.1100	0.6054	2.7737e+06	0.7183	(0.30%)		37158
Total	240.7376 mW	0.6484 mW	5.5846e+06 pW	241.3916 mW			

1

当时钟频率在 200MHz 时，资源消耗情况，预估的面积，功耗情况分别如下所示。

```
Number of ports:                          868
Number of nets:                         51077
Number of cells:                        50144
Number of combinational cells:          35155
Number of sequential cells:             14984
Number of macros/black boxes:               0
Number of buf/inv:                       3252
Number of references:                     107
```

Area

Combinational Area:	657263.379920
Noncombinational Area:	814725.183300
Buf/Inv Area:	26268.581117
Total Buffer Area:	3100.20
Total Inverter Area:	23168.38
Macro/Black Box Area:	0.000000
Net Area:	0.000000
Cell Area:	1471988.563221
Design Area:	1471988.563221

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs	Cell Count
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
clock_network	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
register	194.2908	3.4788e-02	2.8146e+06	194.3284	(99.53%)		14984
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
combinational	0.2238	0.7004	2.5122e+06	0.9266	(0.47%)		35155
Total	194.5146 mW	0.7352 mW	5.3269e+06 pW	195.2551 mW			

当时钟频率在 100MHz 时，资源消耗情况，预估的面积，功耗情况分别如下所示。

```
Number of ports:                          868
Number of nets:                         46704
Number of cells:                        45796
Number of combinational cells:          30807
Number of sequential cells:             14984
Number of macros/black boxes:               0
Number of buf/inv:                       1093
Number of references:                      77
```

Area

Combinational Area:	575324.167736
Noncombinational Area:	859581.690659
Buf/Inv Area:	7836.998502
Total Buffer Area:	991.27
Total Inverter Area:	6845.73
Macro/Black Box Area:	0.000000
Net Area:	0.000000
Cell Area:	1434905.858394
Design Area:	1434905.858394

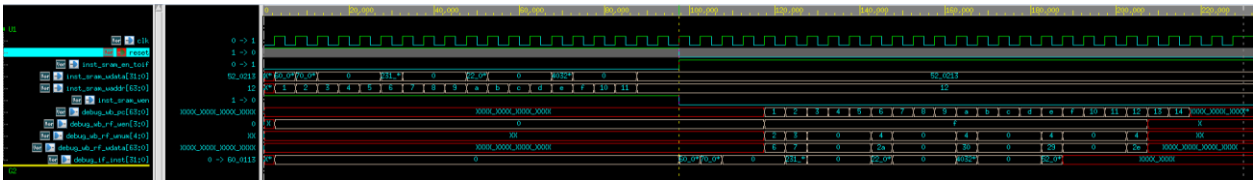
Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs	Cell Count
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
clock_network	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
register	98.7415	1.7755e-02	2.7895e+06	98.7621	(99.64%)		14984
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
combinational	1.9859e-02	0.3305	2.1077e+06	0.3524	(0.36%)		30807
Total	98.7614 mW	0.3482 mW	4.8972e+06 pW	99.1145 mW			

1

五、门级仿真

门级仿真使用 VCS 工具进行编译，使用 Verdi 查看波形结果。门级仿真的结果如下图所示，与 RTL 的仿真结果一致。

门级仿真结果：



RTL 仿真结果：