龙芯杯个人赛设计报告

学校：中山大学
姓名：卢伟聪

一、设计简介

本次个人赛所提交的设计是一个含有 SRAM 控制器、UART 控制器、CPU 内核三大部分的 SOC 系统。其中 CPU 内核是基于 MIPS 指令集的，能够支持 MIPS 的 34 条指令。CPU 的架构采用传统的单发射五级流水线[1]，能够完成三级功能测试的基本要求，并且能够比较快地完成三级性能测试。

为了对 CPU 进行加速，本设计作出了两大尝试。一是尝试在 UART 控制器中引入 FIFO，希望能够对串口进行优化。二是借鉴网上的优秀代码，自行设计了一个 32 位基 4 booth 乘法器。

二、设计方案

（一）总体设计思路

从系统顶层角度出发，整个 SOC 系统分为 SRAM 控制器、UART 控制器、CPU 内核三大部分。

1、CPU 内核

CPU 内核的架构采用传统的单发射五级流水线，因此需要进行取指、译码、执行、访存、写回五大阶段，以及连接各阶段的流水寄存器的设计。同时根据比赛要求，还需要实现寄存器堆，其中有 32 个 32 位的寄存器。为了避免在运行加载存储指令时，对 base_ram 的数据和取指操作冲突，还需要引入流水线暂停机制，因此还需要进行流水线暂停控制模块 control.v 的设计。

（1）取指模块

根据转移标志等信号，决定 pc 在下一个时钟周期的值，并将 pc 输出给 SRAM 控制器，从 base_ram 中取出指令。

（2）译码模块

接收来自取指阶段的信号输入，SRAM 控制器的指令输入，来自执行、访存阶段的数据前推输入，来自寄存器堆的数据输入等。根据输入的指令，用 case 嵌套堆叠[2]的方法，决定输出给执行阶段和寄存器堆的控制信号。并且根据控制信号，确定输出给执行阶段的源操作

数 1、2。

（3）执行模块

接收来自译码阶段的信号输入，根据 aluop 的值，完成相应的子类型运算。再根据 alutype 的值，选择最终的输出结果。其中在执行阶段里尝试引入了电路级乘法器，这是一个 32 位的基 4 booth 乘法器。

（4）访存阶段

一方面，接收来自执行阶段的信号输入，并将信号输出到写回阶段。另一方面，将访存地址、数据、写使能、字节使能等信号输出给 SRAM 控制器和 UART 控制器，通过这些信号对控制器进行数据操作。

（5）写回阶段

接收来自访存阶段的信号输入，并将信号输出到寄存器堆，完成对寄存器堆的写操作。

（6）流水线暂停控制模块

接收来自译码阶段的输入。当译码阶段的指令是加载和存储指令时，流水线暂停控制模块便会发出暂停信号，暂停取指阶段以及取指-译码寄存器。同时模块中的状态机会开始运行，使流水线暂停四个时钟周期，直到加载存储指令完成，再取下一条指令。

2、SRAM 控制器

SRAM 控制器用于控制比赛提供的 base_ram 和 ext_ram 两块 SRAM，其中 base_ram 能够进行数据操作和指令操作，而 ext_ram 只能进行数据操作。

3、UART 控制器

UART 控制器用于控制板上的串口，它可以将 rxd 接收到的数据转换为 32 位数据交给 CPU 使用，也可以将 CPU 的 32 位数据转换为串行数据，通过 txd 发送。

（二）电路级乘法器模块设计

在执行阶段，本设计尝试引入了电路级乘法器。这是一个 32 位的基 4 booth 乘法器[3]。乘法器的设计思路以及各模块的作用为：

1、在 booth_encode.v 文件中，通过乘数的 3 位输入码字，根据 booth 编码表得到相应的输出系数。

2、在 generate_product.v 文件中，通过被乘数和 booth 输出系数，产生不同的部分和。

3、在 wallace_tree.v 文件中，将进位保留加法器 csa 排列成 wallace tree 的形式，将上一步得到的部分和移位相加。并在最后通过行波进位加法器进行最终相加，得到最终的结果。

此模块在执行阶段被调用，它的输入为两个 32 位源操作数的补码，输出为乘法结果。

然而由于时间原因，此乘法器还未能完全完善，仍然存在一些 bug。在进行性能测试时，

若使用此乘法器，只能通过前两个性能测试，无法通过第三个性能测试。

三、设计结果

（一）设计交付物说明

设计的目录层次：

```
├─constrs_1                    #约束文件
│  └─new
├─sim_1                        #仿真文件
│  ├─imports
│  └─new
│      └─include
└─sources_1                    #源代码文件
    ├─ip
    │  ├─fifo_generator_0      #FIFO ip 核
    │  │  ├─doc
    │  │  ├─hdl
    │  │  ├─sim
    │  │  ├─simulation
    │  │  └─synth
    │  └─pll_example           #PLL ip 核
    └─new
        ├─booth                #booth 乘法器
        ├─mycpu                #CPU 内核
        │  ├─ex_stage
        │  ├─id_stage
        │  ├─if_stage
        │  ├─mem_stage
        │  └─wb_stage
        ├─sram                 #SRAM 控制器
        └─uart                 #UART 控制器
```

本设计使用 vivado 2019.2 进行创建，如果要使用，需要先建立好 vivado 工程项目，再将代码添加进工程中。需要注意的是，由于代码是使用 VS Code 编写，其中的中文注释在 vivado 中可能会出现乱码。下面是设计进行仿真、综合、上板演示的必要操作提示步骤：

1、在 vivado 2019.2 中创建工程项目，其中 FPGA 型号选择“xc7a200tfbg676-2”。

2、将代码复制到 src 目录。

3、设置宏定义文件 defines.v 为全局文件，或将其放在每一个子目录中。

4、设置设计中的模板工程文件 thinpad_top 为顶层文件。

5、若要进行行为仿真，需要将指令(.bin)文件的绝对路径复制到 tb.sv 文件的

BASE_RAM_INIT_FILE 中。

6、也可对项目进行综合、生成比特流，从而可以进行上板验证。

（二）设计演示结果

图 1 功能测试 1 的结果

图 2 功能测试 2 的结果

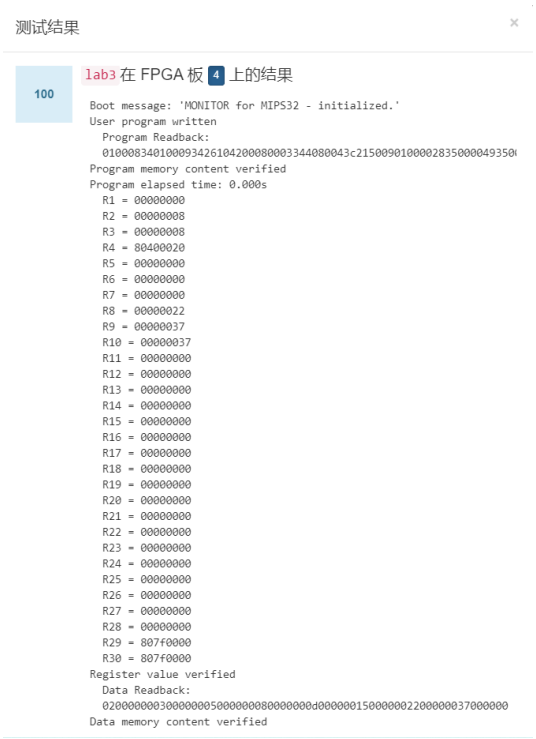

图 3 功能测试 3 的结果

图 4 性能测试 1 的结果

图 5 性能测试 2 的结果

图 6 性能测试 3 的结果

四、参考设计说明

1、CPU 的译码和执行阶段借鉴了雷思磊《自己动手写 CPU》中的 case 思路。

2、SRAM 和 UART 控制器的设计借鉴了 2020 年个人赛的优秀项目 Ge-MIPS 的设计思路。

3、32 位基 4 booth 乘法器的设计，借鉴了网上的优秀代码。

4、FIFO 的设计，调用了 IP 核。

五、参考文献

[1] (美)帕特森(Patterson,D.A),(美)亨尼斯(Hennessy,J.L.).计算机组成与设计:硬件/软件接口(原书第 5 版).
[2] 雷思磊.自己动手写 CPU[M].北京:电子工业出版社,2014 年.
[3] 汪文祥,刑金璋.CPU 设计实战[M].北京:机械工业出版社, 2021 年.