**《计算机组成原理实验》指导手册**

实验简介

1. 实验目标

计算机组成原理实验配合理论课教学，使学生加深理解计算机系统的基本组成结构与工作原理；熟练掌握运用电子设计自动化（EDA）工具、硬件描述语言（Verilog HDL）和可编程逻辑器件（FPGA）设计实现计算机硬件系统的技术和方法；提高学生实践动手能力，以及分析问题和解决问题的能力。

学生从简单的基本部件（ALU和寄存器堆）及其应用设计开始，到逐渐复杂的单周期CPU、多周期CPU和流水线CPU设计，在此基础上再辅以总线与外设接口的设计，最终设计并实现一个真实的（非虚拟或仿真的）虽简单但较为完整的计算机硬件系统。

1. 实验环境和工具

本实验采用与数字电路实验相同的实验环境与工具：Vivado 2016.2、Verilog HDL、Nexsy4-DDR实验板，详细使用说明参见“数字电路实验指导书”。

1. 实验时间和地点

返校前：4月22日开始，周三或周四晚 7:30 ~ 9:30，网上课堂；

返校后：周三或周四晚6:30~9:30，电三楼406、410。

1. 实验课程资源

课程网站：校教务处网络教学平台（https://www.bb.ustc.edu.cn/），课程号：011145.EX.2020SP: 计算机组成原理。

QQ讨论群：2019-2020数字电路-组成原理，293012009。

1. 实验内容及安排
2. 运算器与排序 （1周）
3. 寄存器堆与队列 （1周）
4. 单周期CPU （1周）
5. 多周期CPU （2周）
6. 流水线CPU （3周）
7. 综合设计 （2周）
8. 实验成绩评定

本实验共有六个实验，实验总成绩是六次实验成绩的加权求和。实验四和实验五的权重分别为1.1和1.2，其他为1。每个实验独立评分，满分100分，其中实验检查占80%（包括仿真检查60%和下载检查20%），实验报告占20%。

实验检查时根据实验内容完成情况评分，通过检查演示的实验仿真结果和实验下载后交互运行结果，以及回答问题（例如设计思路、解释代码）情况来评定成绩。实验报告内容包括但不限于：逻辑设计（数据通路和状态图）、核心代码、仿真/下载结果、结果分析、实验总结、意见/建议等。

要求按时完成实验检查和实验报告提交。每次实验检查的截止时间为该次实验规定时长后的晚上9：30，提交实验报告的截止时间是对应实验检查截止时间延后一周。例如第一次实验第10周星期三（4月22日）开始，时长1周，检查截止时间为第11周星期三晚上9：30，报告提交截止时间为第12周星期三晚上12：00。如果拖延造成实验检查或者报告提交时间延迟≤1周，则最多只能得分80%；若延迟≤2周，则最多只能得分60%。延迟超过2周不得分。严禁实验代码和实验报告抄袭，否则作零分处理。

对于按时且超额完成实验的，视超额部分的创意、检查和报告情况，奖励不超过20分（需助教推荐，教师审定）。

1. 复习：Verilog HDL描述注意事项

尽量使用简单、规范的方式描述。

1. 描述组合电路
   * 使用assign或always @\* 描述, “=” 赋值；
   * always描述时避免不完全赋值（否则出现锁存器)；
   * 一定不能包含反馈；
   * 没有必要进行复位。
2. 描述时序电路
   * 使用always @(posedge/negedge clk [,posedge/negedge rst/set])描述, “<=” 赋值；
   * 时钟信号仅出现在边沿敏感变量表中，不应出现在语句块内；
   * 边沿敏感变量表中除时钟和复位/置位信号外，尽量避免其他信号。
3. 变量类型问题
   * initial或always语句中被赋值的变量，未定义直接使用的变量默认为net类型的标量，向量变量必须先定义后使用
   * 同一进程中尽量在一个if或case语句块中对一个变量赋值，否则后边的赋值会覆盖前面的赋值，可能导致逻辑上的问题（特例：组合逻辑描述时，为避免形成锁存器而在开始给变量赋初值）
   * 使用always语句描述的变量，应声明为reg类型（声明为reg类型的变量，综合后不一定生成寄存器）！
   * 使用assign语句描述的变量，应声明为wire类型
4. 多驱动问题
   * 变量在多个always或assign语句中被赋值
   * 模块中所有的assign和always块都是并行执行的，不要在多个并行执行体中对同一变量赋值
5. 多重时钟问题
   * 不能采用行为描述方式来综合实现多个时钟或多个边沿驱动的触发器

always @(posedge clka, posedge clkb)

always @(posedge clk, negedge clk)

一般情况下触发器只有一个时钟且一种边沿驱动

一般不采用时钟上升沿和下降沿的混合设计

1. 参数化模块和规范化代码设计

代码语法采用Verilog HDL1364-2001标准，代码格式规范要求参见文件“Verilog代码规范”。

1. 参数化模块定义格式：

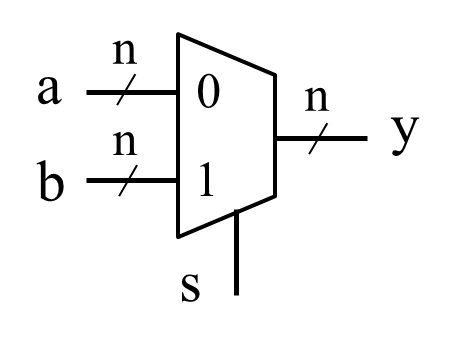
module 模块名 #(parameter 参数声明) (端口声明);

变量声明；

逻辑功能描述；

endmodule

例如：定义2选1数据选择器模块



//参数化的2选1数据选择器模块

module mux2 //模块名：mux2

#(parameter MSB = 31, //参数声明：数据最高有效位编号

LSB = 0 //数据最低有效位编号

)

(output reg [MSB:LSB] y, //端口声明：输出数据

input [MSB:LSB] a, b, //两路输入数据

input s //数据选择控制

);

always @\* //逻辑功能描述

begin

if (s)

begin

y = b;

end

else

begin

y = a;

end

end

endmodule

//另一个更简洁的版本

module mux2 //模块名：mux2

#(parameter WIDTH = 32 //参数声明：数据宽度

)

(output [WIDTH-1:0] y, //端口声明：输出数据

input [WIDTH-1:0] a, b, //两路输入数据

input s //数据选择控制

);

assign y = s? b : a; //逻辑功能描述

endmodule

1. 模块实例化语句格式

模块名 #(参数映射) 例化名 (端口映射);

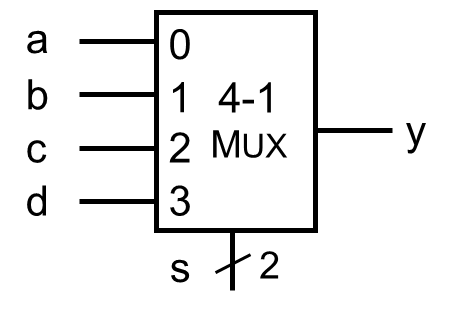
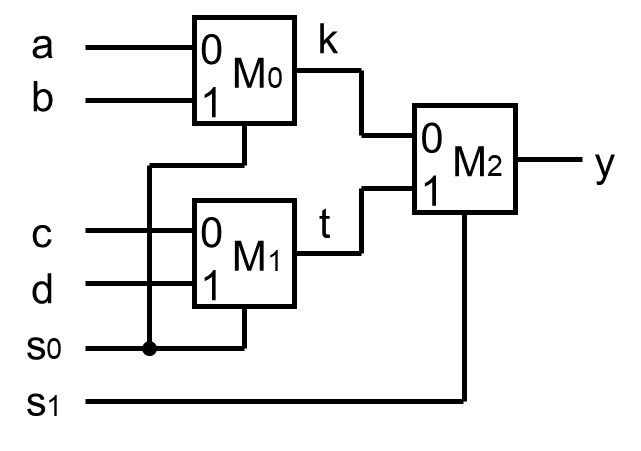
映射方式：基于位置或者基于名字，不可混合使用

位置映射：严格按照模块中声明的顺序传递

名字映射：按照模块中声明的名称对应传递，与顺序无关。格式如下：

**.** 名称 (传入量)

例如：通过例化前面定义的2选1数据选择器模块（mux2）来构造8位4选1数据选择器模块（mux4\_8）。

module mux4\_8

(output [7:0] y,

input [7:0] a, b, c, d,

input [1:0] s

);

wire [7:0] k, t;

//位置映射

mux #(7, 0) M0 (k, a, b, s[0]);

mux #(7) M1 (t, c, d, s[0]);

//名字映射

mux #(.MSB(7)) M2 (.s(s[1]), .a(k), .b(t), .y(y));

endmodule