计算机组成原理(H)

实验简介

2022春季

zjx@ustc.edu.cn

实验简介

- **实验目标：设计实现一个真实（非虚拟或仿真，虽简单但较为完整）的计算机硬件系统**
- **实验工具：Vivado 2019.1，Verilog HDL-2001，Nexys4-DDR开发板或在线实验平台**
- **时间地点：周三或周四晚 6:30~9:30, 电三楼410**
- **课程资源**
  - QQ群：22春_组原实验H
  - VLAB实验中心：vlab.ustc.edu.cn

22春_组原实验H

实验内容

实验一	**运算器及其应用**	**（1周）**
实验二	**寄存器堆与存储器及其应用**	**（1周）**
实验三	**汇编程序设计**	**（1周）**
实验四	**单周期CPU设计**	**（1周）**
实验五	**多周期CPU设计**	**（1周）**
实验六	**流水线CPU设计**	**（2周）**
实验七	**综合设计**	**（3周）**

实验成绩

- **实验总成绩是各次实验成绩的加权和，每次实验成绩包括检查成绩 (80%) 和报告成绩 (20%)**
- **按时完成实验检查和实验报告提交**
  - 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
- **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10分（需教师审定）**

实验检查

- **实验检查内容**
  - 实验仿真结果
  - 实验下载后运行结果
  - 回答问题，例如设计思路、解释代码……

- **实验检查截止时间**
  - 规定时长后的周四晚上9：30

实验报告

- **内容和格式**
  - 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
  - Word或PDF格式，文件名：LabHn_学号_姓名_vi，其中n为第几次实验，vi表示版本号
  - 设计文件附实验报告后，或者以ZIP格式压缩包整合
- **提交网址和截止时间**
  - 网址：vlab.ustc.edu.cn -> 实验报告提交-> H班
    账号：cod2022　　密码：2022cod
  - 截止日期：检查截止时间延后一周的周四晚上12 : 00

复习：Verilog描述注意事项

- **推荐使用简单规范的描述方式**
- **组合电路**
  - 使用assign 或者 always @* 描述, “ = ” 赋值
  - always描述时避免不完全赋值（否则出现锁存器)!
  - 避免出现反馈！例如，y = y + x
  - 无需复位，即组合函数的自变量中无复位信号
- **时序电路**
  - 使用always @(posedge clk, negedge rstn]) 描述, “<=” 赋值
  - 边沿敏感变量表中避免出现除时钟和复位外的其他信号
  - 时钟信号避免出现在语句块内

变量类型问题

- 使用**always**语句描述的变量，务必声明为**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
- **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）

示例：变量类型

```
wire [7:0]  a, b;
reg [7:0]  r1, r2, r3;
```

```
always @(*)        // (en, a, b)
    if (en)  r1 = a;
    else  r1= b;
```

```
always @(en, a)   // @(*)
    if (en)  r2 = a;
```

```
always @(posedge  clk)
    if (en)  r3 = a;
```

组合电路：
敏感变量不要遗漏
所有条件分支均有赋值
不能含有反馈，如r1=r1+1

锁存器：尽量避免使用

寄存器：必有触发时钟

多驱动与多重时钟问题

- **多驱动问题**
  - 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
- **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，例如

    always @(posedge clka, posedge clkb)

    always @(posedge clk, negedge clk)

参数化模块

module 模块名 #(parameter 参数声明) (端口声明);

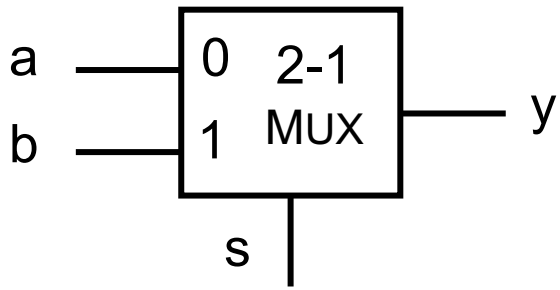
变量声明;

逻辑功能描述;

endmodule

示例：MUX2

```
module mux2                          //模块名：mux2
   #(parameter MSB = 31,             //参数声明：数据最高有效位
   LSB = 0                           //数据最低有效位
   )
   (output [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
   );
assign y = s? b : a;                 //逻辑功能描述
endmodule
```

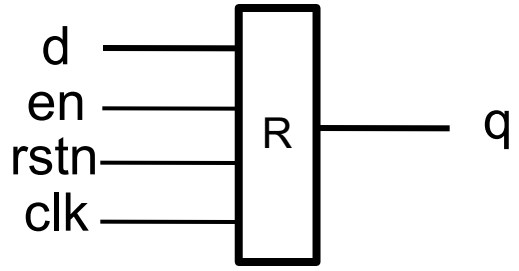


示例：寄存器

```
module  register
   #(parameter WIDTH = 32,
   RST_VALUE = 0)
   (input  clk, rstn, en,
   input  [WIDTH-1 : 0]  d,
   output reg  [WIDTH-1 : 0] q);

always @(posedge clk, negedge rstn)
   if (!rstn) q <= RST_VALUE;
   else  if (en)
      q <= d;

endmodule
```



- d, q：输入、输出数据
- clk, rstn, en：时钟、复位、使能

寄存器功能表

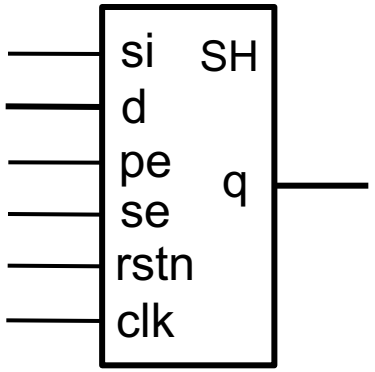
rstn	clk	en	q	功能
0	x	x	0	复位
1	↑	1	d	置数
1	↑	0	q	保持

示例：移位寄存器

```
module  shifter
    #(parameter N = 8,
    RST_VALUE = {N{1'b0}})
    (input  clk, rstn, pe, se,
    input  [N-1: 0]  d,
    output reg  [N-1: 0] q);

always @(posedge clk, negedge rstn)
    if (!rstn) q <= RST_VALUE;
    else if (pe) q <= d;
    else if (se) q <= {si, q[N-1: 1]};

endmodule
```



移位寄存器功能表

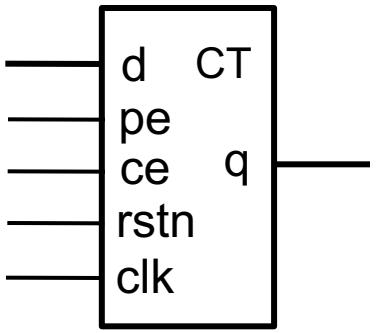
rstn	clk	pe	se	功能
0	x	x	x	复位
1	↑	1	x	置数
1	↑	0	1	右移
1	↑	0	0	保持

示例：计数器

```
module  counter
    #(parameter N = 8,
    RST_VALUE = {N{1'b1}})
    (input  clk, rstn, pe, ce,
    input  [N-1: 0]  d,
    output reg  [N-1: 0] q);

always @(posedge clk, negedge rstn)
    if (!rstn) q <= RST_VALUE;
    else if (pe) q <= d;
    else if (ce) q <= q-1;

endmodule
```



递减计数器功能表

rstn	clk	pe	ce	功能
1	x	x	x	复位
0	↑	1	x	置数
0	↑	0	1	计数
0	↑	0	0	保持

模块实例化

- **模块实例化语句格式：**

  module_name #(parameter_map) instance_name (port_map);

- **端口映射方式：基于位置或者基于名字，不可混合使用**

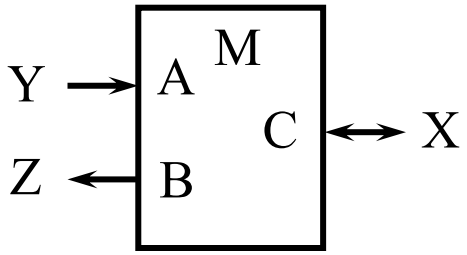
  – 位置映射：按模块中端口定义的顺序传递

  例如：模块定义为 moduel M(A, B, C);

  M M1(Y, Z, X);　　//顺序很重要

  – 名字映射：.PortName (value)

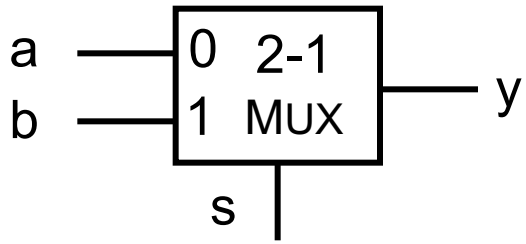
  M M1(.B(Z), .C(X), .A(Y));　// 顺序无关

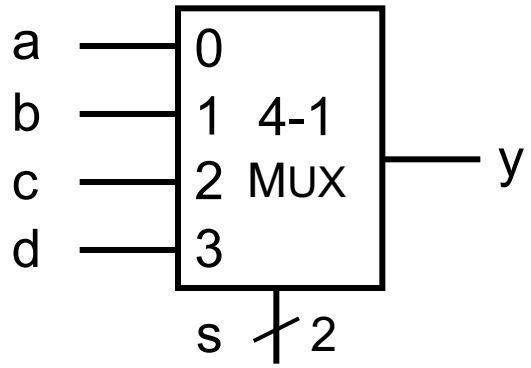


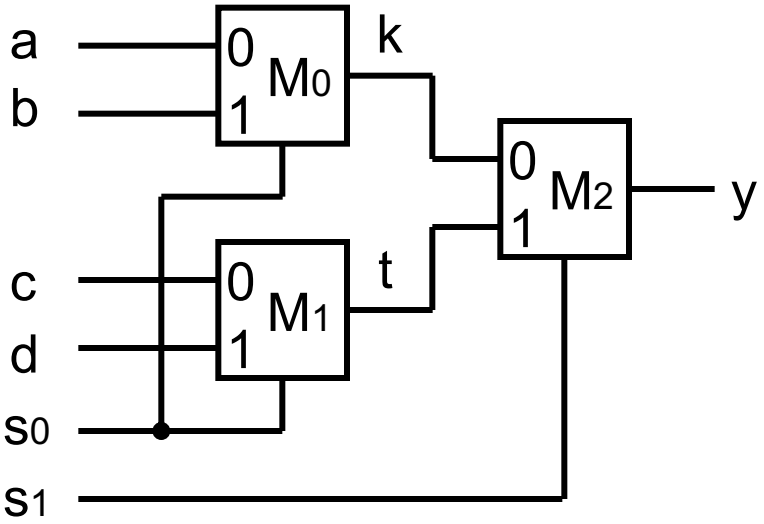
- **参数的映射方法类似**

示例：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
mux2 #(7, 0) M0 (k, a, b, s[0]);
mux2 #(7) M1 (t, c, d, s[0]);
//名字映射
mux2 #(.MSB(7)) M2 (.s(s[1]), .a(k),
     .b(t), .y(y));
endmodule
```

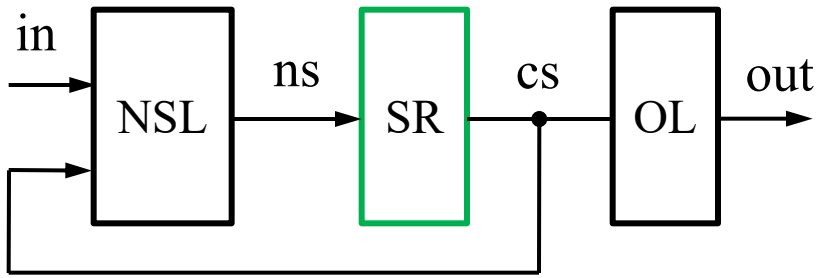






FSM描述

- **Moore型FSM的两段式Verilog描述**




```
// 描述cs
always  @(posedge clk, negedge rstn) begin
    if (!rstn)   cs <= S0;       //异步复位
    else   cs <= ns;
end
```

```
//描述ns和out
always  @*  begin
    ns = cs;
    out = 默认值;
    case  (cs)
        S0: begin
            out = 表达式0;
            if (in条件0) ns = Si;
            else if ……
            ……
        end
        S1: begin
        .....
    endcase
end
```

仿真时钟

```
reg clk;
// 时钟周期和个数
parameter CYCLE = 10, Number = 20;

initial  begin
  clk = 0;
  repeat  (2* Number ）//或者 forever
      # CYCLE/2  clk = ~ clk;
end
```

- **initial和#仅用于仿真，不会产生实际硬件电路**

The End