**湖 北 大 学**

计算机与信息工程学院

2023 —— 2024学年度

第 一 学期

**学 生 实 验 报 告 册**

学生姓名：

班 级： 通信工程2101班

学 号： 202131116030035

课程名称： EDA应用设计

任课老师： 周艳玲

学生实验守则

1、学生在规定的时间内进行实验，不得无故缺席或迟到。

2、学生在每次实验前对排定要做的实验应进行预习，并按要求作好预习报告。

3、每次实验前，必须交上次实验报告和本次实验预习报告，并经指导教师提问、检查同意后，才可进行本次实验。

4、学生进入实验室指定位置后，首先根据仪器清单核对自己使用的仪器是否有缺少或损坏，发现问题及时向指导教师报告，严禁擅自动用别组仪器。

5、实验时必须有实事求是、严肃认真的科学态度，严格遵守仪器操作规程和注意事项。

6、实验完毕应将实验数据交给指导教师检查，合格后，整理复原好仪器设备，方可离开实验室。

7、保持实验室肃静和整洁，不得大声喧哗，乱丢垃圾和吃东西。

8、学生在实验过程中，由于不遵守操作规程或未经许可，擅自进行实验而造成事故、损坏仪器设备，应及时报告，并填写损坏清单，按院有关规定进行赔偿。

实 验 报 告 单

**实验名称： 频率计**

同组人 无 实验室 实训大楼618-619 时间2023

**一、实验目的：**

1.了解Verilog 语法：：generate 相关语法、for 循环、阻塞赋值（=）

和非阻塞赋值（<=）的区别。

2.学习典型时序电路逻辑：状态机、多位十进制计数器。

**二、实验内容：**

建立一个频率计电路，用于测量输入波形的频率。该电路采用直

接测量法实现简单频率测量，间接测量法和等精度测量法等方法读者

可尝试自行设计。

该电路为简单时序电路，包含顶层模块、状态机模块、多位十进

制计数器模块，以及第2.3 节中介绍的分频模块、第2.4 节中介绍的

数码管扫描显示模块和第2.5 节中介绍的计数器模块等。

状态机模块产生系统所需控制信号，主要包括计数清除、计数使

能、计数锁存等。顶层模块在计数使能条件下进行计数生成固定时间

为1s 的闸门信号，并产生计数溢出信号。计数溢出信号指示闸门信

号结束，状态机据此进行状态跳转，产生计数锁存信号。将闸门信号

区间内对待测输入信号的计数值锁存，并通过数码管显示其高4 位十

进制数（单位10KHz）。随后产生计数清除信号，对相关计数器的计

数值进行清楚，准备进行下一次频率测量。

顶层模块包含产生待测信号的逻辑，分别对27MHz 的输入时钟信

号进行2、4、6、8 分频并输出到指定引脚。在下载测试时，可分别

将这些引脚与待测信号输入引脚相连，验证频率测量结果是否正确。

**三、实验代码：**

dec\_counter 参数化十进制计数器模块

------------------------------------------------------------------------------------------------------

`timescale 1ns/1ns

module dec\_counter#(

parameter DIG\_WIDTH = 8

)(

input clk,

input rst,

input enable\_i,

output cout\_o,

output [DIG\_WIDTH\*4-1:0] count\_o

);

wire [DIG\_WIDTH:0] cout;

assign cout[0] = clk;

assign cout\_o = cout[DIG\_WIDTH];

genvar i;//定义循环变量 产生八个counter实例

generate

for(i=0;i<DIG\_WIDTH;i=i+1) begin : CNT\_GEN

counter #(

.WIDTH (4),

.DATA\_MAX (9)

) counter\_i0 (

.clk (cout[i]),

.rst (rst),

.pulse\_i (enable\_i),

.cout\_o (cout[i+1]),

.count\_o (count\_o[4\*(i+1)-1:4\*i])//把输入时钟分频位1/2^8

);

end

endgenerate

endmodule

------------------------------------------------------------------------------------------------------

state\_ctrl 状态控制模块

------------------------------------------------------------------------------------------------------

module state\_ctrl(

input clk,

input rst,

input cout\_i,

output reg clear\_o,

output reg count\_en\_o,

output reg lock\_o

);

localparam SIM\_DELAY =1;

localparam ST\_IDLE=2'h0;

localparam ST\_CLR=2'h1;

localparam ST\_CNT=2'h2;

localparam ST\_LOCK=2'h3;

wire cout;

reg [1:0] curr\_state;

reg [1:0] next\_state;

//时钟触发的状态更新块

always @(posedge clk or posedge rst) begin

if(rst) begin

curr\_state <= #SIM\_DELAY ST\_IDLE;

end else begin

curr\_state <= #SIM\_DELAY next\_state;

end

end

always @(\*) begin

case(curr\_state)//确定下个状态的值

ST\_IDLE: next\_state=ST\_CLR;

ST\_CLR: next\_state=ST\_CNT;

ST\_CNT: next\_state=cout\_i?ST\_LOCK:ST\_CNT;

ST\_LOCK: next\_state=ST\_IDLE;

default: next\_state=ST\_IDLE;

endcase

end

always @(posedge clk or posedge rst) begin

if(rst)begin

clear\_o = #SIM\_DELAY 1'b0;

count\_en\_o = #SIM\_DELAY 1'b0;

lock\_o = #SIM\_DELAY 1'b0;

end else begin

clear\_o = #SIM\_DELAY 1'b0;

count\_en\_o = #SIM\_DELAY 1'b0;

lock\_o = #SIM\_DELAY 1'b0;

case(curr\_state)

ST\_CLR:clear\_o = #SIM\_DELAY 1'b1;

ST\_CNT:count\_en\_o = #SIM\_DELAY 1'b1;

ST\_LOCK:lock\_o = #SIM\_DELAY 1'b1;

endcase

end

end

endmodule

------------------------------------------------------------------------------------------------------

freq\_meter\_demo

------------------------------------------------------------------------------------------------------

module freq\_meter\_demo(

input clk,

input rst\_n,

output [3:0]clk\_out\_o,

input clk\_in\_i,

output [3:0]dig\_o,

output [7:0]seg\_o

);

localparam CLK\_FREQ\_Hz = 27000000;

wire rst=~rst\_n;

genvar i;

generate

for(i=0;i<4;i=i+1) begin :FREQ\_DIV\_GEN

freq\_div #(

.DIV\_RATE\_2N ((i+1)\*2) //分频2 4 6 8

) freq\_div\_i (

.clk\_in\_i (clk),

.rst\_i (rst),

.clk\_out\_o (clk\_out\_o[i])

);

end

endgenerate

wire cout\_1s;

wire clear;

wire count\_en;

wire lock;

state\_ctrl state\_ctrl\_i(

.clk (clk),

.rst (rst),

.cout\_i (cout\_1s),

.clear\_o (clear),

.count\_en\_o (count\_en),

.lock\_o (lock)

);

counter #(

.WIDTH ($clog2(CLK\_FREQ\_Hz)),

.DATA\_MAX (CLK\_FREQ\_Hz-1)

) counter\_i0 (

.clk (clk),

.rst (rst|clear),

.pulse\_i (count\_en),

.cout\_o (cout\_1s),

.count\_o ()

);

wire [8\*4-1:0] freq;

dec\_counter #(

.DIG\_WIDTH (8)

)(

.clk (clk\_in\_i),

.rst (rst|clear),

.enable\_i (count\_en),

.cout\_o (),

.count\_o (freq)

);

reg [15:0] freq\_reg;

always @(posedge clk or posedge rst) begin

if(rst) begin

freq\_reg[15:0] <=16'h0;

end else if(lock) begin

freq\_reg[15:0] <=freq[31:16];

end

end

seg\_ctrl #(

.DIG\_WIDTH (4)

) seg\_ctrl\_i (

.clk (clk),

.rst (rst),

.data\_i (freq\_reg[15:0]),

.dig\_o (dig\_o[3:0]),

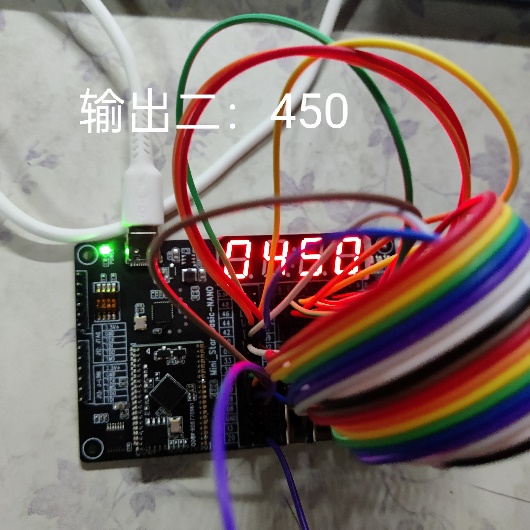
.seg\_o (seg\_o[7:0])

);

endmodule

**四、实验结果：**

显示四个频率值





成绩：

批阅教师：————————

日 期：————————