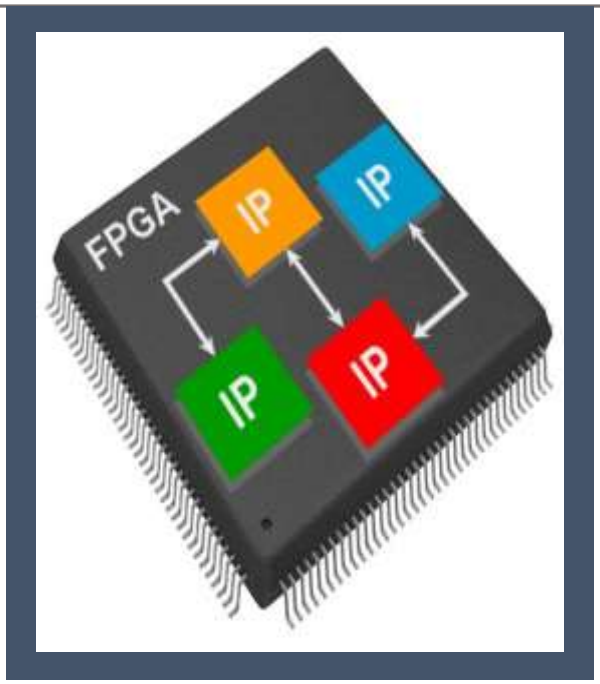


EDA 设计基础

指导手册(2019)

xilingsnow@163.com

目 录

上机一：EDA 工具运行和工程调试

一、实验目的

1、掌握 Quartusll 设计、综合和调试的基本流程

2、掌握 modelsim 仿真工具的输入、仿真的基本流程

3、掌握 iverilog 和 gtkwave 编译和仿真工具的使用方法和流程。

4、了解基本数字系统开发的步骤和技术

二、预习要求

1、了解数字逻辑系统分析和设计的方法、流程。

2、了解 Quartusll 软件安装和破解流程，保证软件可靠运行

3、了解 iverilog 和 gtkwave 编译和仿真命令。

4、了解原理图和文本输入的主要特性和区别。

5、了解开发板（DE2-35/115）的核心器件的型号和参数。

三、实验原理和步骤

1、建立工程，选择合适的文件夹保存工程文件。

2、选择合适的 FPGA 目标器件，根据目标板的核心器件选择。

3、逻辑设分析设计：三人表决器：三人表决，以少数服从多数为原则，多数人同意则议案通过，否则议案被否决。这里，我们使用三个波动开关代表三个参与表决的人，置“0”表示该人不同意议案，置“1”表示该人同意议案；两个指示灯用来表示表决结果，LEDR0 点亮表示议案通过，LEDR1 点亮表示议案被否决原理图设计方式，真值表如下。采用原理图和文本实现方式。

（1）、原理图方式，采用基本的与或非门实现电路设计。

（2）、HDL 方式，分别设计三人表决电路模块代码和测试模块代码。

S1	S2	S3	LED1	LED 2
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1

1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

4、选择原理图方式，建立原理图逻辑连接图，并对工程进行综合、调试。

5、测试：

（1）、建立波形向量文件（vwf），对工程进行测试。并观测运行结果。

（2）、设计 Verilog HDL 模块文件和测试文件，分别利用 modelsim 和 gtkwave 进行仿真测试，并观测运行结果。

四、实验内容

1、参照附录中 QuartusII 的使用方法，按照实验原理，设计一个三人表决器，进行原理图设计、综合，并利用向量波形文件（vwf）进行功能仿真。

2、参照附录中 QuartusII 和 modelsim 的使用方法。采用 Verilog HDL 设计三人表决电路模块。采用 verilog HDL 编写表决器的测试模块，并利用 Modelsim 进行仿真，观察波形查看结果。

3、参照 iverilog&gtkwave 使用方法。对测试文件进行适当修改，使用 gtkwave 进行仿真，观察波形输出结果。

五、实验要求

1、按照上面的实验内容，进行设计分析、原理图或代码文本设计，进行综合、仿真。

2、将程序编译过程中出现的主要错误及解决方案进行总结

3、掌握和练习数字系统的分析步骤，并将一些核心步骤进行记录

4、将编写的程序与仿真结果进行记录

上机二：VERILOG HDL 基本模块设计

一、实验目的

1、熟练掌握文本输入设计、综合、仿真和下载的基本流程

2、掌握 Verilog HDL 模块的构成和基本逻辑模块的设计规范。

3、掌握测试和错误修改方法。

二、预习要求

1、了解 EDA 数字系统设计的基本流程。

2、了解 QuartusII 或 iverilog&gtkwave 软件使用的基本规范和操作流程

3、了解原理图和文本输入的主要特性和区别。

三、实验原理

1、Verilog HDL 模块的结构模板。

```
module   <顶层模块名> (<输入输出端口列表>);
output   输出端口列表;              //输出端口声明
input    输入端口列表;              //输入端口声明
/*定义数据，信号的类型，函数声明*/
reg   信号名;
//逻辑功能定义
assign <结果信号名>=<表达式>;   //使用 assign 语句定义逻辑功能
always @ (<敏感信号表达式>) //用 always 块描述逻辑功能
   begin
    //过程赋值
    //if-else，case 语句
    //while，repeat，for 循环语句
    //task，function 调用
   end
//调用其他模块
 <调用模块名 module_name > <例化模块名> (<端口列表 port_list >);
//门元件例化
 门元件关键字  <例化门元件名> (<端口列表 port_list>);
endmodule
```

2、三裁判电路设计。a、b、c 表示三位裁判，out 表示输出，同意输出 1，不同意输出 0。参考代码如下。

```
module lab001(a,b,c,out);//模块定义。
input a,b,c;
```

```
output    out;
wire[1:0] sum;//sum 变量的用途传递计算和，注意和范围为 0-3，所以设定为 2 位
assign sum=a+b+c;//assign 语句，再 always 模块外部，独立语句必须用 assign
assign out=sum[1]?1:0;//条件运算符，思考 sum[1]代表的含义
endmodule
```

3、三裁判电路的 always 设计。参考代码如下。

```
module lab002(a,b,c,out);
input a,b,c;
output out;
reg[1:0] sum;
assign sum=a+b+c;
always@(a,b,c)//always 过程定义，注意上面 assign 和 always 过程块（begin-end 之间）是并行结构，并行运行。注意 always 括号内信号列表的格式和含义
begin
   if(sum>=2) out=1;//if 语句对 out 进行赋值，注意此处不用 assign
   else out=0;
end
endmodule
```

4、三裁判电路，采用模块调用设计。参考代码如下。

```
module lab003(a,out);//主模块
input[2:0] a;
output    out;
wire[1:0] out1;
lab0031 lab1(a,out1);//模块调用
assign    out=(out1>=2)?1:0;//assign 语句、always 过程块、模块调用，三种传值结构是完全并行，可以多次使用，但是不能够嵌套。Lab0031 模块产生的输出值，通过中间变量 out1 传递到 assign 语句中，参与处理。
endmodule

module lab0031(a,out);//被调用子模块
input[2:0] a;
output[1:0] out;//注意此处为什么为 2 位。
assign out=a[0]+a[1]+a[2];//注意此处为位选择方式，可以对多位变量 a 进行逐位访问。
endmodule
```

3、2 位加法器，ina，inb 为两个 2 位的操作数，sum 为相加的和，count 为进位。

2 位加法器　　四种不同的实现方法，注意左右表格对比	

module　add(ina,inb,sum,count);
input[1:0] ina,inb;
output[1:0] sum;
output count;
	module add (ina,inb,sum,count);
	input[1:0] ina,inb;
	output reg[1:0] sum;//注意不同
	output reg count;//注意不同

wire[2:0] sum_temp;
assign sum_temp=ina+inb;
assign sum=sum_temp[1:0];
//或 assign sum=ina+inb;
assign count=sum_temp[2];
endmodule
	reg[2:0] sum_temp;//注意不同
	always@(ina,inb)//组合逻辑用 ina、inb 电平形式作为驱动信号
	begin
	sum_temp=ina+inb;
	sum=sum_temp[1:0];//或 sum=ina+inb;
	count=sum_temp[2];
	endmodule

module add (ina,inb,sum,count); input[1:0] ina,inb; output reg[1:0] sum; output reg count; always@(ina,inb) begin {count,sum}=ina+inb; endmodule	module add (ina,inb,sum,count); input[1:0] ina,inb; output[1:0] sum; output count; assign {count,sum}=ina+inb;//{ }为位拼接运算符，可以实现二进制数据、参量的拼接 endmodule

4、带优先级 8-3 编码器和 3-8 译码器

带优先级 8-3 编码器

```
module code83(in8,out3);
input[7:0] in8;     output reg[2:0] out3;
always@(in8)
begin
 case(in8)
  if(in8[7]) out3=3'b111;
  else if(in8[6]) out3=3'b110;
        else if(in8[5]) out3=3'b101;
              else if(in8[4]) out3=3'b100;
                    else if(in8[3]) out3=3'b011;
                          else if(in8[2]) out3=3'b010;
                                else if(in8[1]) out3=3'b001;
                                      else if(in8[0]) out3=3'b000;
                                            else   out3=3'bxxx;
 endcase
end
endmodule
```

3-8 译码器

```
module decode38(in3,out8);
input[2:0] in3;   output[7:0] out8;
always@(in3)
begin
 case(in3)
  3'd0:out=8'b00000001;
  3'd1:out=8'b00000010;
```

```
    3'd2:out=8'b00000100;
    3'd3:out=8'b00001000;
    3'd4:out=8'b00010000;
    3'd5:out=8'b00100000;
    3'd6:out=8'b01000000;
    3'd7:out=8'b10000000;
    default:out=8'bxxxxxxxx;
  endcase
end
endmodule
```

四、实验内容

1、实验原理中七裁判电路，可以用 assign 语句、always 过程、模块调用实现。参考模块模板，了解 Verilog HDL 模块的基本结构，了解模块调用的规范。

2、参照 lab001、lab002、lab003 三种不同的设计方式，实现 7 裁判电路设计，并建立 vwf 测试文件进行测试，查看测试结果是否符合逻辑要求。并进行适当扩展，比如增加裁判数量，比如 10 个，11 个，如果偶数个裁判的话，5 比 5 的情况如何处理，输出情况除了通过、不通过或待定等状态。

3、设计编码显示电路：现在由 7----0（优先级高到低）八路输入（高电平 1 为有效电平），输出 3 位编码和 1 位输出有效状态信号。

比如 第 7 路输入，则输出 111 编码，输出状态为 1（表示输出有效），

.....

第 0 路输入，输出 000，输出状态为 1

没有输入，输出为 000，输出状态为 0

将有效输出的编码 111-000 以十进制形式显示到七段管上（7---0）。

无效输出 000，七段管不亮。

4、二进制数显示电路：将 8 个拨动开关作为 8 位并行输入，即对应数码为 0000_0000---1111_1111,十进制数值为 0-255，将编码对应的十进制数值显示在三个七段管上。

5、8 位二进制乘法：两个 8 位二进制数 a，b，c=a+b；将输入 a，b 和结果 c 显示到七段管上。

6、在文件代码综合测试过程中，总结错误原因，掌握错误定位和改进的方法。

五、实验要求

1、按照上面的提示完善报告

2、将程序编译过程中出现的主要错误及解决方案进行总结

3、掌握和练习数字系统的分析步骤，并将一些核心步骤进行记录

4、将编写的程序与仿真结果进行记录

上机三：基本时序逻辑设计

一、实验目的

1、熟练掌握时序逻辑设计规范

2、掌握 Verilog HDL 时序逻辑模块的设计和仿真。

3、掌握开发板上基本输出资源的使用方法。

3、掌握测试和错误修改方法。

二、预习要求

1、了解 EDA 数字系统设计的基本流程。

2、了解时序逻辑过程块设计的基本规范和模型

3、了解七段管的工作特性。

3、了解组合逻辑和时序逻辑电路设计的主要特性和区别。

三、实验原理

1、SW 上拨为输出高电平、下拨为输出低电平

2、KEY0-KEY3 正常情况下输出高电平，按下输出低电平，松开按键恢复到弹起状态输出高电平

3、7 段数码管：HEX0[6:0]---HEX7[6:0]，每个 7 段管信号对应数码管，6---0，低电平时发光，高电平不亮。

4、分频电路参考：由 50MHz 信号，经过分频变成 1Hz。原理：0-25000000 计数，满 2500 万后新的时钟 CLK1 反转，计数归零，从 0 开始重新计数到 2500 万后，新时钟再反转，使得 CLK1 一个周期内包含 2500 万*2 个 CLOCK_50（50Mhz）时钟周期，为 1hz

```
module divclk1hz(reset,clk50,clk1);

input clk50,reset;//clk50 为输入 50Mhz 信号，reset 为复位信号

output reg clk1;//新产生的 1hz 信号

integer i=0;//50Mhz 频率下，周期计数器

always@(posedge clk50)

begin
```

```
if(!reset)    begin i=0;end

else    begin

    if(i==25000000)    begin i=0; clk1=~clk1;end

    else i=i+1;

    end

end

endmodule
```

四、实验内容

1、字符移位：在 8 个 7 段管上显示 HELLO_ _ _(可以显示下划线或不亮也可)，每隔 1 秒钟，字符序列左移或右移一个七段管的位置。系统外部时钟 50Mhz。左/右移位可以通过一个波动开关 sw0 来控制。

2、计时器：在 6 个七段管上分别显示 小时（0-23 或 11）、分（0-59）、秒（0-59），各占 2 个管。外部时钟 50Mhz。可以用按键来产生一个复位信号 key，当按键按下立刻（异步）将时间复位成 0 小时、0 分、0 秒重新开始计时

3、扩展：将时分秒计时器，变成可调的计时器。可以分别对小时、分、秒设定初始值。设定的方法可以采用，从外部输入一个值，比如小时可以从 5 开始；也可以通过按键把小时当前的值增加或者减少来实现值的调整。

4、在文件代码综合测试过程中，总结错误原因，掌握错误定位和改进的方法。

五、实验要求

1、按照上面的提示完善报告

2、将程序编译过程中出现的主要错误及解决方案进行总结

3、掌握和练习数字系统的分析步骤，并将一些核心步骤进行记录

4、将编写的程序与仿真结果进行记录

上机四：FSM 时序电路设计

一、实验目的

掌握时序逻辑设计思路

掌握状态机实现数字逻辑设计的方法和基本形式

掌握状态机的设计注意事项

二、预习要求

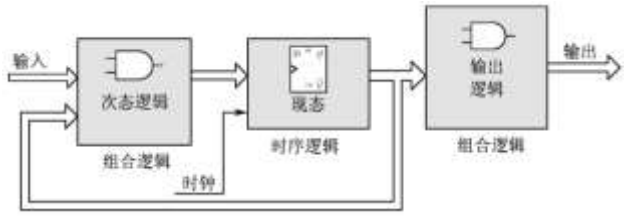
1.了解 veirlogHDL 行为语句。

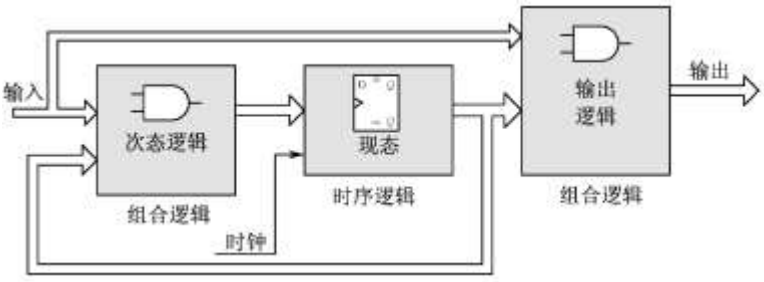
2. Verilog HDL 的模块结构的组成。

3.状态机设计方法。

三、实验原理

1、莫尔型状态机和米利型状态机原理图





2、FSM 设计模 5 计数器代码

```
module fsm(clk,clr,z,qout);

input clk,clr; output reg z; output reg[2:0] qout;

always @(posedge clk or posedge clr) //此过程定义状态转换

begin if(clr) qout<=0; //异步复位

else case(qout)
```

3'b000: qout<=3'b001;

3'b001: qout<=3'b010;

3'b010: qout<=3'b011;

3'b011: qout<=3'b100;

3'b100: qout<=3'b000;

default: qout<=3'b000; /*default 语句*/

endcase

end

always @(qout) /*此过程产生输出逻辑*/

begin case(qout)

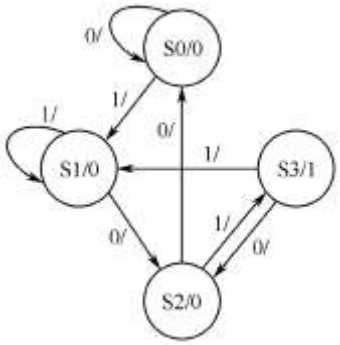
3'b100: z=1'b1;

default:z=1'b0;

endcase

end

endmodule



3、FSM 设计 101 序列检测器

三过程实现	单过程实现

module fsm1_seq101(clk,clr,x,z);
input clk,clr,x; output reg z; reg[1:0] state,next_state;
parameter S0=2'b00,S1=2'b01,S2=2'b11,S3=2'b10;
/*状态编码，采用格雷（Gray）编码方式*/
always @(posedge clk or posedge clr) /*该过程定义当前状态*/
beginif(clr) state<=S0; //异步复位，s0 为起始状态
else state<=next_state;
	module fsm4_seq101(clk,clr,x,z);
	input clk,clr,x; output reg z; reg[1:0] state;
	parameter S0=2'b00,S1=2'b01,S2=2'b11,S3=2'b10;
	/*状态编码，采用格雷（Gray）编码方式*/
	always @(posedge clk or posedge clr)
	Begin if(clr) state<=S0; //异步复位，s0 为起始状态
	else case(state)
	S0:begin if(x) begin state<=S1; z=1'b0;end
	else begin state<=S0; z=1'b0;end

```
end
always @(state or x) /*该过程定义次态*/
begin
case (state)
S0:begin if(x) next_state<=S1;
else next_state<=S0; end
S1:beginif(x) next_state<=S1;
else next_state<=S2; end
S2:begin
if(x) next_state<=S3;
else next_state<=S0; end
S3:beginif(x) next_state<=S1;
else next_state<=S2; end
default:next_state<=S0; /*default 语句*/
endcase
end
always @(state) /*该过程产生输出逻辑*/
begin case(state)
S3: z=1'b1;
default:z=1'b0;
endcase
end
endmodule
```

```
end
S1:begin if(x) begin state<=S1; z=1'b0;end
else begin state<=S2; z=1'b0;end
end
S2:beginif(x) begin state<=S3; z=1'b0;end
else begin state<=S0; z=1'b0;end
end
S3:begin if(x) begin state<=S1; z=1'b1;end
else begin state<=S2; z=1'b1;end
end
default:begin state<=S0; z=1'b0;end
/*default 语句*/
endcase
end
endmodule
```

四、实验内容

1、参考实例代码，设计一个 4 连续 0 或者 4 个连续 1 的序列检测 FSM，定义一个长序列，在七段管上分别显示检测的 4 个连续 0 和 4 个连续 1 的个数。显示连续 0 和连续 1 的个数在七段管上的显示，分别用函数和任务实现。

2、移动速度可控的 HELLO 自动循环显示，当 KEY1 按下后，循环速度为每秒移动 1 个七段管的位置，按下 KEY0 后，速度减慢，4 秒移动一个七段管的位置。用 FSM 设计实现。

五、实验要求

1、按照上面的提示完善报告

2、将程序编译过程中出现的主要错误及解决方案进行总结

3、掌握和练习数字系统的分析步骤，并将一些核心步骤进行记录

4、将编写的程序与仿真结果进行记录

实验一:CPU 指令运算器设计

一、实验目的

（1） 掌握 QuartusII 等实验工具的输入、综合、仿真、下载的使用方法

（2） 掌握 DE2 开发版的器件功能特性和使用方法

（3） 掌握 Verilog HDL 组合逻辑系统设计的主要方法和技术

（4） 掌握并应用设计的方法和流程

二、预习要求

（1） 了解 QuartusII 等管教分配、下载的方法和流程

（2） 了解开发板输入、输出显示资源的工作特性

（3） 了解开发板设计、开发和测试的方法和流程

三、实验要求

设计一个简单的 CPU 指令运算器，指令格式如下。

操作类型 1	操作数 1

操作类型 2	操作数 3	操作数 4

完成的具体功能定义如下：

（1） 操作类型 1：将操作数 1 作为一个无符号二进制数，在七段管以十进制显示二进制序列等效值。

（2） 操作类型 2：实现操作数 3、操作数 4 之间相加、减、乘的操作，在七段管以十/十六进制进制显示操作数和结果。操作数 3 和 4 为 BCD 码表示的 2 位十进制数（表示的值为 00-99）。

注意：

（1） 操作类型 2 中，减法逻辑中出现负数，则显示“-”，正数可以不显示符号

（2） 操作类型 2 中，加、减、乘操作数和结果都用十进制显示，可以在七段管上进行循环显示来实现。

（3） 注意操作数 3、4 以 BCD 码输入，超过 9 的 BCD 码输出处理问题。

（4） 尝试加法运算采用流水线方式实现。注意有效位数。

（5） 如果感觉七段管显示能力弱，可以查询 LCD1602 的控制模块代码，采用 LCD 显示。

实验二:功能可调综合计时器设计

一、实验目的

（1） 掌握 QuartusII 等实验工具的输入、综合、仿真、下载的使用方法

（2） 掌握 DE2 开发版的器件功能特性和使用方法

（3） 掌握 Verilog HDL 时序逻辑系统设计的主要方法和技术

（4） 掌握并应用 EDA 设计的方法和流程

二、预习要求

（1） 了解 QuartusII 等管教分配、下载的方法和流程

（2） 了解开发板输入、输出显示资源的工作特性

（3） 了解开发板设计、开发和测试的方法和流程

三、实验要求

设计一个可调的综合计时器。具体功能：

（1） 显示小时、分、秒，提供置零功能。显示在七段管或 LCD 屏幕上，可以考虑 24/12 小时模式切换功能。

（2） 能够对秒、分、小时进行分别修改，可以两位数整体修改或每位独立修改

（3） 整点报时功能，整点可以显示一定形式的 LED 或 LCD 屏幕上来表示。

（4） 闹钟功能，设定特定时间，到时间以特定 LED 或 LCD 屏幕上显示来显示闹钟。注意闹钟持续时间，也可以参考懒人闹钟模式。

实验三:局域网数据帧检测 FSM 电路设计（选做）

一、实验目的

（1） 掌握 Quartusll 等实验工具的输入、综合、仿真、下载的使用方法

（2） 掌握 DE2 开发版的器件功能特性和使用方法

（3） 掌握 Verilog HDL 的复杂 FSM 系统设计的主要方法和技术

（4） 掌握并应用 EDA 设计的方法和流程

二、预习要求

（1） 了解 Quartusll 等管教分配、下载的方法和流程

（2） 了解开发板输入、输出显示资源的工作特性

（3） 了解开发板设计、开发和测试的方法和流程

三、实验要求

设计一个帧检测系统，设定帧的格式如下：

01110	数据 1	01110

01110 序列作为一个帧的定界符号，通过检测前后“01110”定界符，确定帧的长度，获取数据 1。数据 1 设定最大长度 10bit，将数据 1 的无符号二进制序以十进制输出。

注意：

（1） 数据 1 可以设定为固定长度，则检测到开头的“01110”后取固定长度的序列就可以。

（2） 数据 1 可以设定为可变长度，需要检测开头和结尾的定界符，确定数据 1 的有效长度。

（3） 注意在测试时候，数据 1 中不能出现“01110”序列，防止定界差错。也可采用数据处理，比如采用比特填充方法（查询：“帧比特填充”资料自行学习）。