Verilog 练习一　基本逻辑门和组合电路的设计

一．练习内容

1. 用 Verilog 完成三态门的设计，并用 ModelSim 进行仿真。

2. 用 Verilog 完成 4 位加法器的设计，并使用 ModelSim 进行仿真。

二．源程序及仿真结果截图

1. 三态门

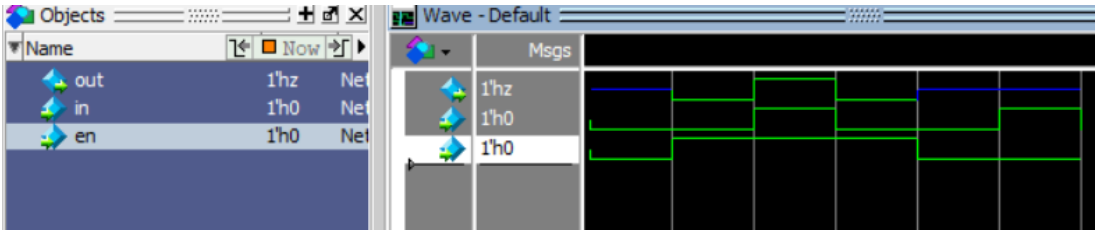
实现代码：

```
//三态门
module tri_gate1(out,in,en);
    output out;
    input in,en;
    assign out = en?in:1'bz;
    //若 en = 1,out = in;若 en = 0, out 为高阻态
endmodule
```

仿真代码：

```
module tri_tb;

    reg    en  ;
    reg    in  ;
    wire   out ;

    tri_gate1 uut(
        .en(en),
        .in(in),
        .out(out)
    );

    initial begin
        en = 0;
        in = 0;
        #10 en = 1;
        #10 in = 1;
        #10 in = 0;
        #10 en = 0;
        #10 in = 1;
        #10 in = 0;
    end

endmodule
```

仿真结果截图



2.四位加法器

实现代码

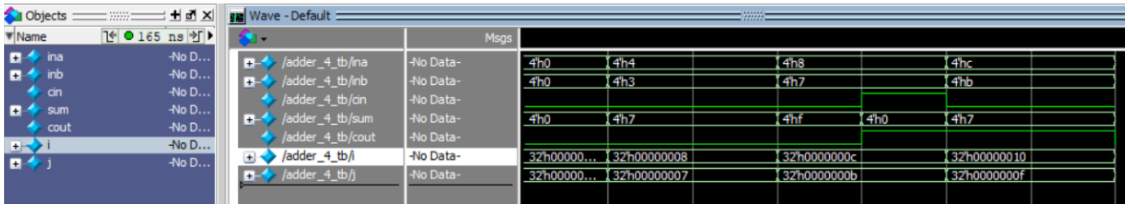
```
module adder_4(cout,sum,cin,ina,inb);

    output[3:0] sum      ;
    output      cout     ;
    input[3:0]  ina,inb  ;
    input       cin      ;

    assign {cout,sum} = ina + inb + cin;

endmodule
```

仿真代码

```
module adder_4_tb;

reg[3:0]  ina,inb;
reg       cin    ;
wire[3:0] sum    ;
wire      cout   ;
integer   i,j    ;

adder_4 uut(
    .ina(ina),
    .inb(inb),
    .cin(cin),
    .sum(sum),
    .cout(cout)
);

initial begin
    ina = 0;
    inb = 0;
    cin = 0;
end

initial begin
    for(i = 4;i<=16;i = i + 4)
        #40 ina = i;
```

```
end

initial begin
    for(j = 3;j<=16;j = j + 4)
        #40 inb = j;
end

initial begin
    #100 cin = 1;
    #20  cin = 0;
end

endmodule
```

仿真结果截图



Verilog 练习二　组合电路的设计

一．练习内容

1. 用 Verilog 完成 3-8 译码器的设计，并用 ModelSim 进行仿真。

2. 用 Verilog 完成两个 4 位二进制数据比较器的设计，使用 ModelSim 进行仿真。

二．源程序及仿真结果截图

1. 3-8 译码器

实现代码

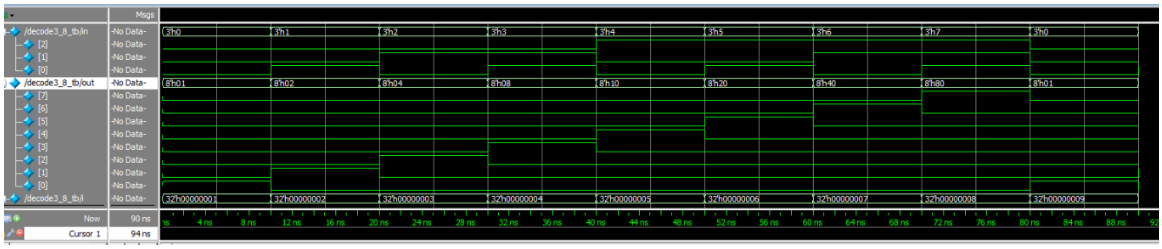
```
module decode3_8(out,in);

input[2:0] in  ;
output[7:0]   out ;
reg[7:0]  out ;

always @(in)
   begin
      case(in)
         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'b00000000;
      endcase
   end

endmodule
```

仿真代码

```
module decode3_8_tb;

reg[2:0]  in  ;
wire[7:0] out ;
integer   i   ;

decode3_8 uut(
   .in(in),z
```

```
    .out(out)
);

initial begin
    in = 0;
    for(i = 1; i<=9; i = i + 1)
        #10 in = i;
end

endmodule
```

仿真结果截图



2.两个二进制数据比较器设计

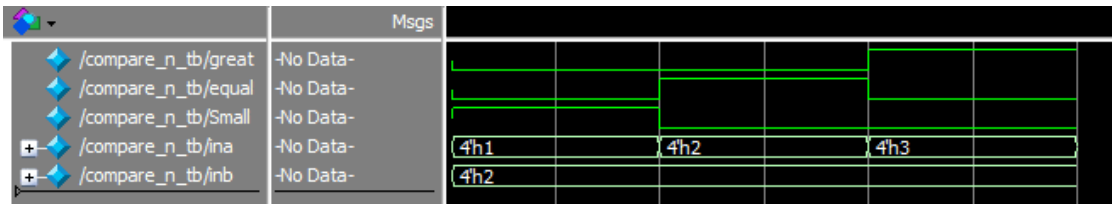
实现代码

```
module compare_n (great, equal, Small,ina,inb);
    parameter  n=4;
    output  great, equal, Small;
    input[n:1]   ina,inb;

    assign great = (ina>inb);
    assign equal = (ina==inb);
    assign Small = (ina<inb);
/*
    assign great = (ina[3]>inb[3])
        ||((ina[3]==inb[3])&&(ina[2]>inb[2]))
        ||((ina[3]==inb[3])&&(ina[2]==inb[2])&&(ina[1]>inb[1]))
        ||((ina[3]==inb[3])&&(ina[2]==inb[2])&&(ina[1]==inb[1])&&(ina[0]>in
b[0]));

    assign equal = (ina[3]==inb[3])
        &&(ina[2]==inb[2])
        &&(ina[1]==inb[1])
        &&(ina[0]==inb[0]);

    assign Small = ~(great^equal);
*/
endmodule
```

仿真代码

```
module compare_n_tb;

parameter     n=4;
wire       great, equal, Small  ;
reg[n:1]   ina,inb              ;
integer    i                    ;

compare_n uut(
    .great(great),
    .equal(equal),
    .Small(Small),
    .ina(ina),
    .inb(inb)
);

initial begin

    ina = 1;
    inb = 2;

    for(i = 0;i < 3;i = i + 1)
        #10 ina = ina + 1;
end

endmodule
```

仿真结果截图



Verilog 练习三　触发器

一．练习内容

1. 用 Verilog 完成具有异步清零和异步置 1 的 D 触发器的设计，并用 ModelSim 进行仿真。

2. 用 Verilog 完成时钟下降沿触发的 JK 触发器的设计，使用 ModelSim 进行仿真。

二．源程序及仿真结果截图

1．异步清零和异步置 1 的 D 触发器

实现代码：

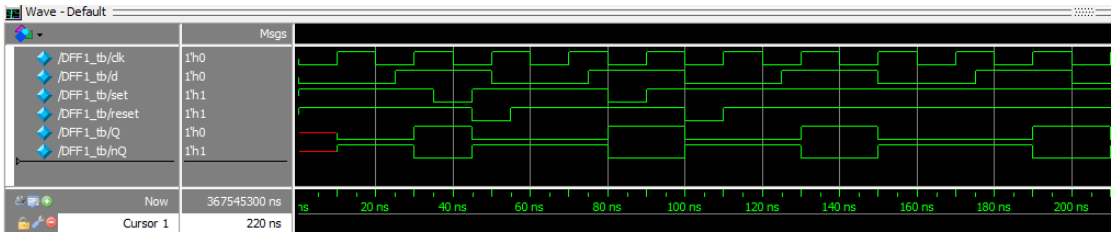
```
module DFF1(clk,d,set,reset,Q,nQ);

input      clk,d,set,reset;
output reg Q,nQ;

always @(posedge clk or negedge set or negedge reset) begin

    if(!reset) begin
        Q <= 0;
        nQ <= 1;
    end
    else if(!set) begin
        Q <= 1;
        nQ <= 0;
    end
    else begin
        Q <= d;
        nQ <= ~d;
    end

end

endmodule
```

仿真代码：

```
module DFF1_tb;

reg    clk,d,set,reset;
```

```
wire   Q,nQ;

DFF1 uut(
    .clk(clk)     ,
    .d(d)         ,
    .set(set)     ,
    .reset(reset) ,
    .Q(Q)         ,
    .nQ(nQ)
);

initial begin
    clk = 0;
    forever begin
        #10 clk = ~clk;
    end
end

initial begin
    d = 0;
    forever begin
        #25 d = ~d;
    end
end

initial begin
    set = 1;
    #35 set = ~set;
    #10 set = ~set;

    #35 set = ~set;
    #10 set = ~set;
end

initial begin
    reset = 1;
    #45 reset = ~reset;
    #10 reset = ~reset;

    #45 reset = ~reset;
    #10 reset = ~reset;
end

endmodule
```

仿真结果截图



2. 时钟下降沿触发的 JK 触发器

实现代码

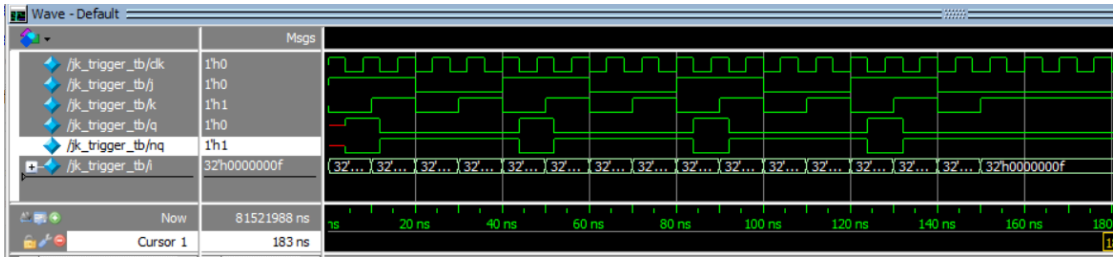
```
module jk_trigger(j,k,q,nq,clk);

input      j,k,clk;
output reg q,nq;

always @(negedge clk) begin
    if(!clk) begin
        if(j == 0 && k == 1) begin
            q <= 0;
            nq <= 1;
        end
        else if(j == 1 && k == 0)begin
            q <= 1;
            nq <= 0;
        end
        else if(j == 1 && k == 1)begin
            q <= ~q;
            nq <= ~nq;
        end
    end
end

endmodule
```

仿真代码

```
module jk_trigger_tb;

reg    j,k,clk;
wire   q,nq;
integer    i;

jk_trigger uut(
    .j(j),
    .k(k),
    .clk(clk),
    .q(q),
```

```
    .nq(nq)
);

initial begin
    clk = 1;
    k = 0;
    j = 1;
    for(i = 0;i < 15;i = i + 1)
        #10 {j,k} = {j,k} + 1;
end

always #4 clk = ~clk;

endmodule
```

仿真结果截图



Verilog 练习四 计数器

一．练习内容

1. 用 Verilog 完成具有异步清零和异步置 1 的模 10 计数器的设计，并用 ModelSim 进行仿真。

2. 用 Verilog 完成具有同步清零和同步步置 1 的模 16 计数器的设计，使用 ModelSim 进行仿真。

二．源程序及仿真结果截图

1．异步清零和异步置 1 的模 10 计数器

实现代码：

```
module mod10_counter(cnt,cf,en,clk,clr,rst);

parameter     size = 4      ;
parameter     mod = 10      ;
input         en,clr,clk,rst    ;
output        cf            ;
output[size:1]    cnt           ;
reg[size:1]       cnt           ;
reg           cf,cf_ff      ;

always @(posedge clk or negedge clr or negedge rst) begin
    if(!clr) begin
        cnt = 0;
        cf = 0;
    end
    else if(!rst)
        cnt = 1;
    else if(cnt == (mod-1) )
        cnt = en?0:cnt;
    else
        cnt = cnt + en;
end

always  @(posedge clk)begin     //进位标志
    if(cnt == (mod-1))begin
```

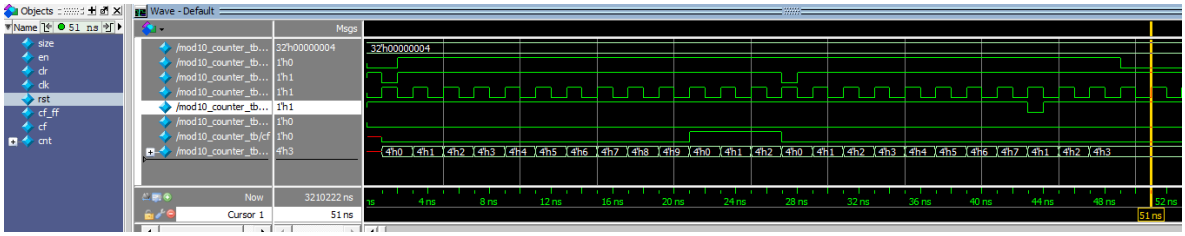
```
        cf_ff <= 1;
    end
    else begin
        cf_ff <= cf_ff;
    end
    cf = (cf == 1)?1:(cnt == 0)&en&(cf_ff == 1) ;
end

endmodule
```

仿真代码：

```
module mod10_counter_tb;

parameter     size = 4             ;
reg           en,clr,clk,rst,cf_ff ;
wire          cf                   ;
wire[size:1]  cnt                  ;

mod10_counter uut(
    .cnt(cnt),
    .cf(cf),
    .en(en),
    .clk(clk),
    .clr(clr),
    .rst(rst)
);

initial begin
    en = 0;
    clk = 0;
    clr = 1;
    rst = 1;
    cf_ff = 0;

    #1
    clr = ~clr;
    #1
    clr = ~clr;
    en = ~en;     //en = 1
    #25
    clr = ~clr;       //清零检验
    #1
    clr = ~clr;
    #15
    rst = ~rst;       //置数检验
```

```
    #1
    rst = ~rst;
    #5
    en = ~en;      //en = 0
end

always #1 clk = ~clk;

endmodule
```

仿真结果截图



2. 同步清零和同步置 1 的模 16 计数器

实现代码

```
module mod16_counter(cnt,cf,en,clk,clr,rst);

parameter          size = 4       ;
parameter          mod = 16       ;
input              en,clr,clk,rst;
output             cf             ;
output[size:1]     cnt            ;
reg[size:1]        cnt            ;
reg                cf,cf_ff       ;

always @(posedge clk) begin     //计数器
    if(!clr) begin
        cnt <= 0;
        cf <= 0;
        cf_ff <= 0;
    end
    else if(!rst)
        cnt = 1;
    else if(cnt == (mod-1) )
        cnt = en?0:cnt;
    else
        cnt = cnt + en;
end

always  @(posedge clk)begin     //进位标志
    if(cnt == (mod-1))begin
```

```
        cf_ff <= 1;
    end
    else begin
        cf_ff <= cf_ff;
    end
    cf = (cf == 1)?1:(cnt == 0)&en&(cf_ff == 1) ;
end

endmodule
```

仿真代码

```
module mod16_counter_tb;

parameter     size = 4          ;
reg           en,clr,clk,rstcf_ff  ;
wire          cf                ;
wire[size:1]  cnt               ;

mod16_counter uut(
    .cnt(cnt),
    .cf(cf),
    .en(en),
    .clk(clk),
    .clr(clr),
    .rst(rst)
);

initial begin
    en = 0;
    clk = 0;
    clr = 1;
    rst = 1;
    cf_ff = 0;

    #1
    clr = ~clr;
    #2
    clr = ~clr;
    en = ~en;     //en = 1
    #70
    clr = ~clr;       //清零检验
    #2
    clr = ~clr;
    #14
    rst = ~rst;       //置数检验
```

```
    #2
    rst = ~rst;
    #5
    en = ~en;      //en = 0
end

always #2 clk = ~clk;

endmodule
```

仿真结果截图