

实验05 使用 vivado 进行仿真

【实验目的】

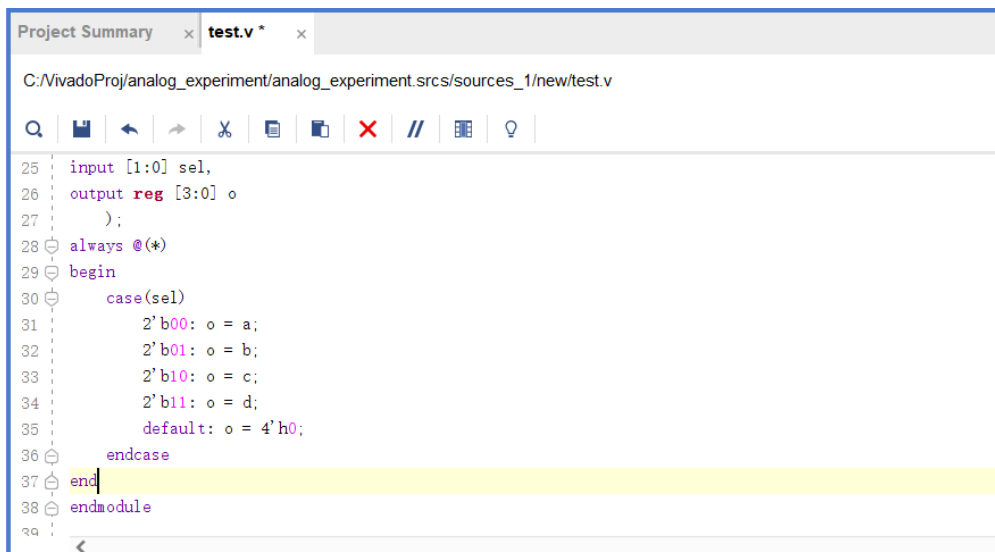
- 熟悉 Vivado 软件的下载、安装及使用
- 学习使用 Verilog 编写仿真文件
- 学习使用 Verilog 进行仿真，查看并分析波形文件

【实验环境】

- PC 一台
- < www.vlab.ustc.edu.cn >
- Vivado 工具

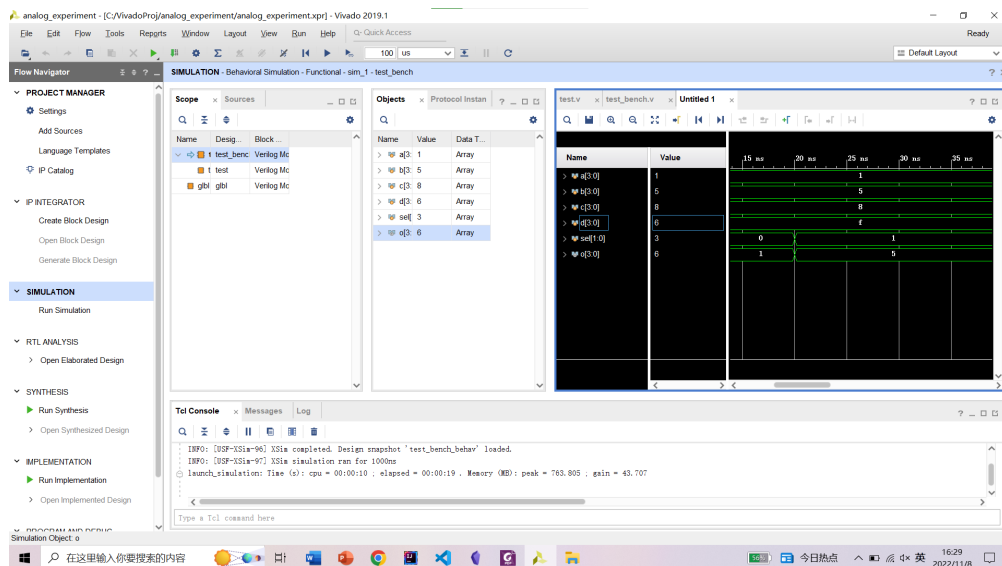
【实验过程】

按照实验指导手册中的指示逐步完成软件下载，工程的创建，添加设计文件等步骤。
输入示例代码得到：

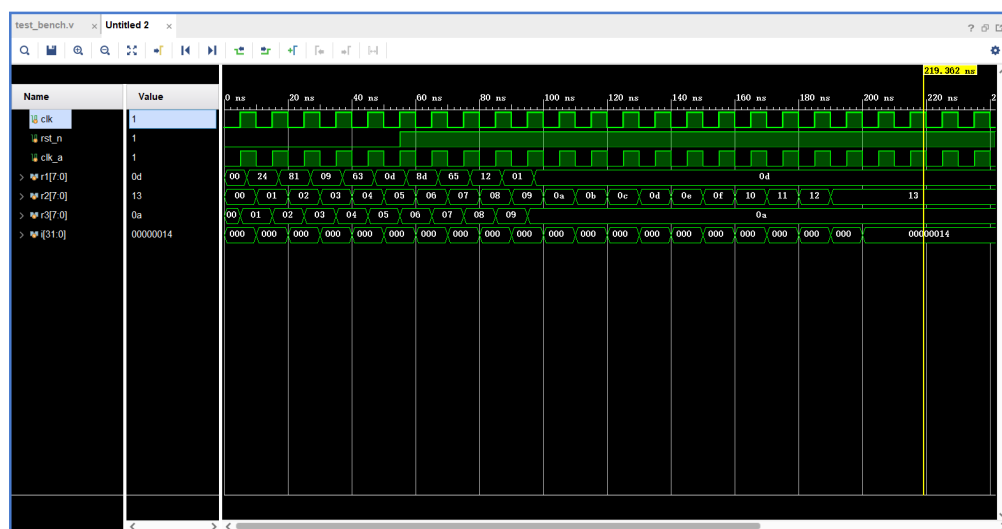


```
25 input [1:0] sel,
26 output reg [3:0] o
27 );
28 always @(*)
29 begin
30     case(sel)
31         2'b00: o = a;
32         2'b01: o = b;
33         2'b10: o = c;
34         2'b11: o = d;
35         default: o = 4'h0;
36     endcase
37 end
38 endmodule
```

在 vivado 中添加 verilog 仿真测试文件，并按照指示进行仿真。

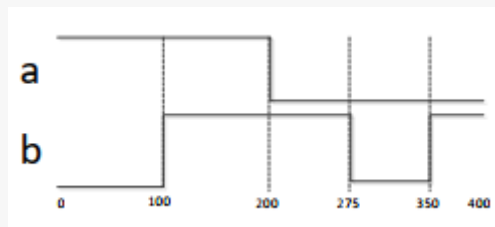


进行第二个测试文件的仿真，进一步学习并熟悉一些有关语法。



【实验练习】

题目 1. 请编写 Verilog 仿真文件，生成如下图所示的波形，并在 Vivado 中进行仿真。



根据题目所示波形编写如下仿真文件：

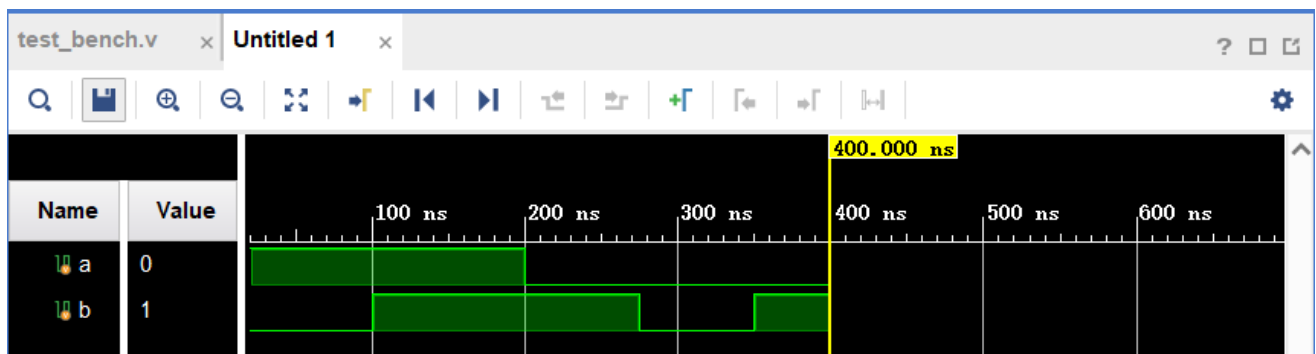
```
module test_bench();
    reg a, b;
    initial begin
        a = 1;
        b = 0;
    end
endmodule
```

```

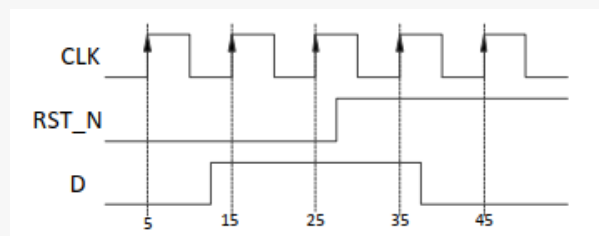
        #100 b = 1;
        #100 a = 0;
        #75 b = 0;
        #75 b = 1;
        #50 $stop;
    end
endmodule

```

在 vivado 软件中将该仿真文件添加到项目中，启动仿真模拟得到如下与题目要求相匹配的波形图：



题目 2. 请编写 Verilog 仿真文件，生成如下图所示的波形，并在 Vivado 中进行仿真。



根据题目所示波形编写如下仿真文件：

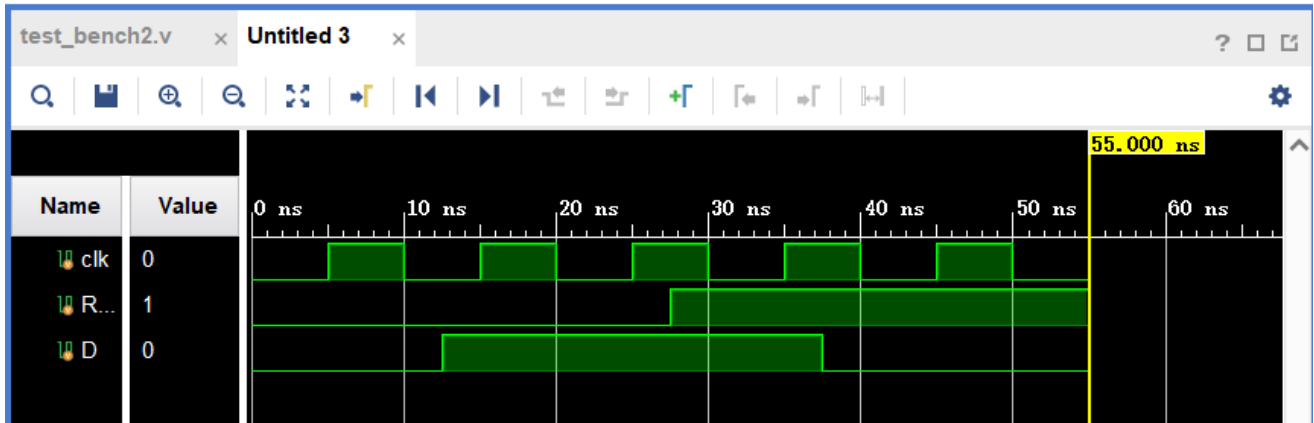
```

module test_bench2();
    reg clk, RST_N, D;
    initial begin
        clk = 0;
        forever #5 clk = ~clk;
    end
    initial begin
        RST_N = 0; D = 0;
        #12.5 D = 1;
        #15 RST_N = 1;
        #10 D = 0;
        #17.5 $stop;
    end
endmodule

```

```
end
endmodule
```

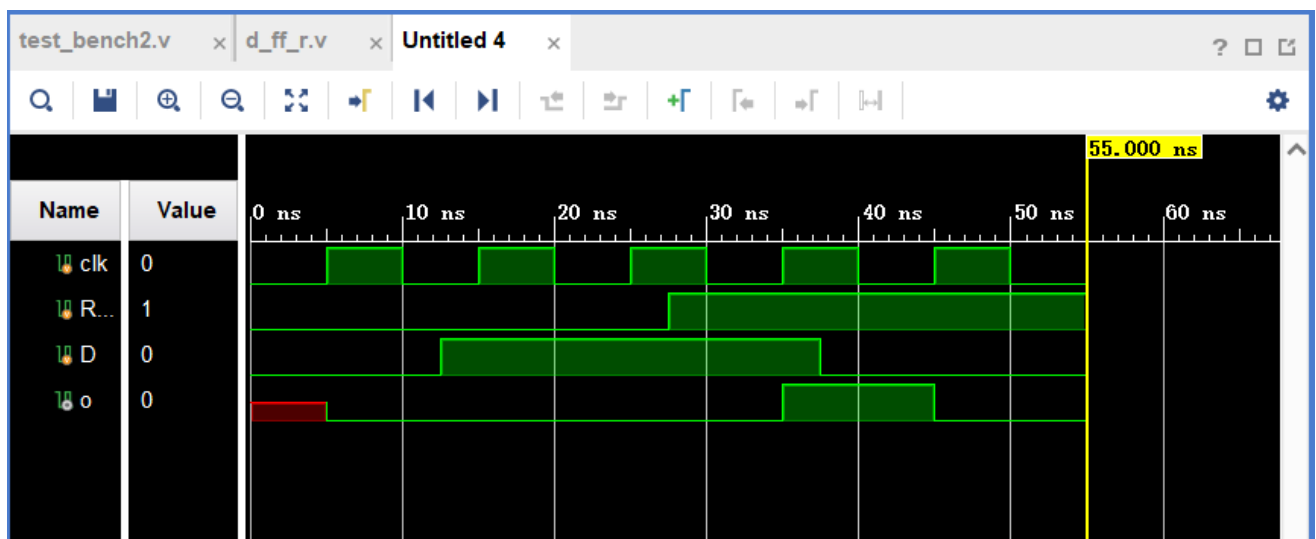
在 vivado 软件中将该仿真文件添加到项目中，启动仿真模拟得到如下与题目要求相匹配的波形图：



题目 3. 利用题目 2 中的信号作为以下代码的输入，在 Vivado 中对其仿真，并观察仿真波形。

```
module d_ff_r(
input clk,rst_n,d,
output reg q);
    always@(posedge clk)
    begin
        if(rst_n==0)
            q <= 1'b0;
        else
            q <= d;
        end
    end
endmodule
```

将上述代码作为源文件添加到 t2 的项目中，并在 t2 代码的基础上加上在 t3 中会使用到的端口。进行仿真模拟得到如下波形：



观察波形可知，第一个时钟上升沿来临前 o 值未知，RST_N 无效，后在每个时钟上升沿 o 更新为 D 的值。

题目 4. 设计一个 3-8 译码器，编写仿真测试文件，在 Vivado 中对其进行仿真。要求仿真时遍历所有的输入情况组合，给出源代码和仿真截图。

3-8 译码器有三个使能端，其中 E1, E2 为低电平有效，E3 为高电平有效，有任何一个使能无效时，输出全为 1。

当使能有效时，输出为 3 位二进制输入所对应的十进制数，相应十进制数所对应的输出端口置 0。

由此可以得到 3-8 译码器的 verilog 代码如下：

```
module decode(
    input [3:0] A,
    input _E1, _E2, E3,
    output reg [7:0] _Y
);
always @ (*)
begin
    if (E3 == 1'b0 || E2 == 1'b1 || E1 == 1'b1) //使能无效
        _Y = 8'b1111_1111;
    else //使能有效
        begin
            case (A)
                3'h7: _Y = 8'b0111_1111; //输出最高位Y7有效
                3'h6: _Y = 8'b1011_1111;
                3'h5: _Y = 8'b1101_1111;
                3'h4: _Y = 8'b1110_1111;
                3'h3: _Y = 8'b1111_0111;
                3'h2: _Y = 8'b1111_1011;
```

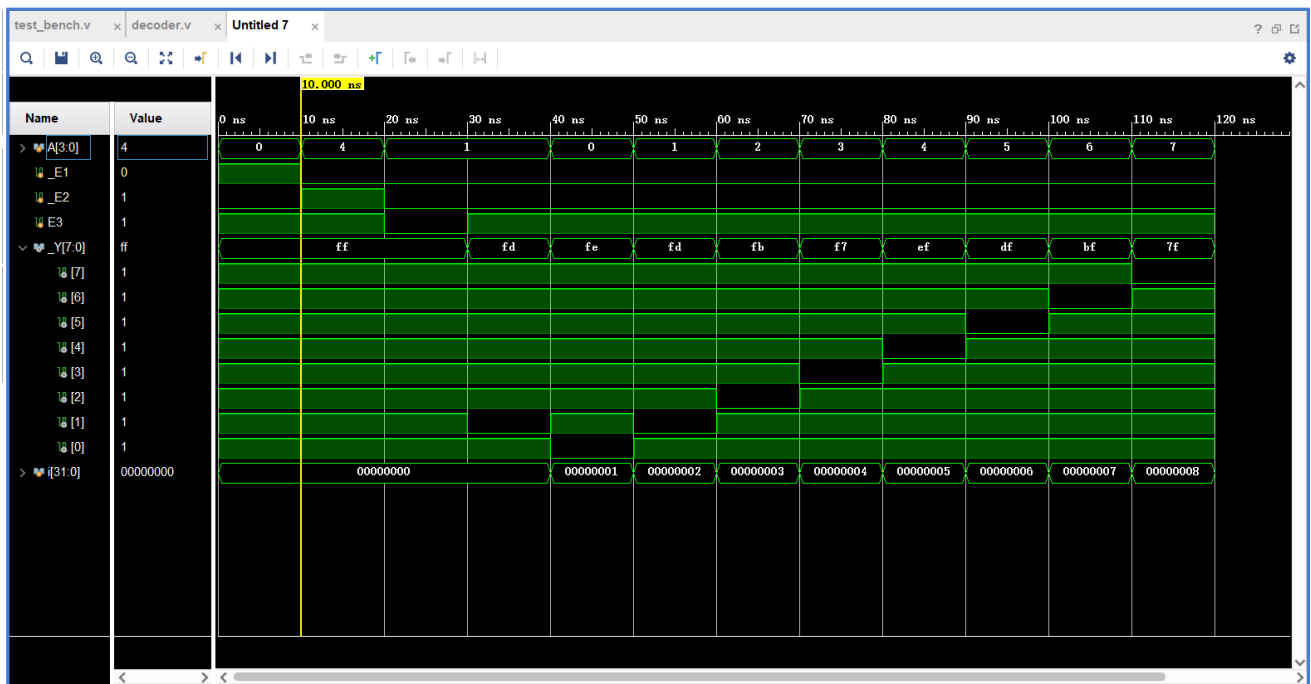
```

3'h1: _Y = 8'b1111_1101;
3'h0: _Y = 8'b1111_1110;
default: _Y = 8'b1;
endcase
end
end
endmodule

```

为了遍历所有情况，仿真测试先使使能无效，此时输入 A 为 0~7 的随机值；再使使能有效，先测试一组随机输入，再按顺序测试。

将以上两文件按照 t2, t3 的方式添加到同一项目中进行模拟仿真，得到以下波形图：



解释：前三段为使能无效，第四段使能有效且输入一随机值，第五段开始依次输入 7~0。

【总结与思考】

1. 实验收获

- 学习了如何使用 vivado 软件进行对 verilog 代码的仿真
- 掌握了在生成仿真文件时所需要的 verilog 语法
- 能够根据波形图对模块进行分析和错误排查

2. 难易程度

- 难度适中，主要难点在熟练使用 vivado 工具和熟悉运用特殊 verilog 语法上

3. 任务量

- 任务量偏多，主要在熟悉工具和看明白语法上花了大量时间，实际编写题目耗时不多

4. 对本次实验的建议

- 指导书上对于 vivado 的使用介绍的不够详尽，比如同一项目加入两个文件，一个调用另一个输出信号作为输入具体的操作原理的步骤；调整时间、保存波形图的方法

这样的细节没有提到，还要花时间自己摸索

- 同时对于新接触的语法也建议给出更多示例帮助学生更好地理解