实验3 简单时序电路设计

* 任务描述
* 相关知识
* 实验内容
* 遇到的问题及解决方法
* 实验心得、意见和建议

## 任务描述

1. 掌握Verilog语言的简单时序电路的设计、实现、仿真、调试方法。

2. 掌握锁存器、触发器、简单寄存器、移位寄存器和计数器等器件的建模和使用，了解这些器件带复位、使能、加载等功能的用法。

3. 掌握用测试平台（test bench）对模块进行测试和验证的方法。

4. 通过仿真波形图分析所设计模块功能的正确性。

## 相关知识

设计中经常用到时序电路，为保证时序正确，需要进行时序控制。时序控制可以与过程语句关联，时序控制有延迟控制和事件控制两种形式。

(1) 延迟控制

格式为：*#delay 过程语句*

比如： #10 Q = 4'b1001; 表示等待10个时间单位后执行赋值。

(2) 事件控制

事件控制又分跳变沿敏感事件控制和电平敏感时间控制。所谓跳变沿是指信号由低电平变为高电平（上升沿）或由高电平变为低电平（下降沿）的那一瞬间。

跳变沿敏感事件控制格式为： *@event 过程语句*

比如， @(posedge clock) curr\_state = next\_state; 表示在clock信号上出现了正跳变沿（上升沿），就执行赋值语句；否则，赋值语句被挂起。负跳变沿事件的表示是在信号前面加negedge，比如，@(negedge clock) 表示clock信号出现负跳变沿的事件。

事件控制中的敏感事件可由多个表达式组成，用or或逗号把它们隔开，形成敏感事件列表。@ \* 表示隐含地把过程语句中所有变量和线网都包含在敏感事件列表中。

## 实验内容

1、锁存器和触发器是时序电路中常用的存储器件。下面分别给出了D锁存器和D触发器（时钟上升沿触发）的行为建模。

module D\_latch(input clk, input D, output reg Q);

always @ (clk or D)

if (clk) begin

Q <= D;

end

endmodule

module D\_ff(input clk, input D, output reg Q);

always @ (posedge clk) // 时钟上升沿触发

Q <= D;

endmodule



左图电路中，从上到下分别是D锁存器、时钟上升沿触发的D触发器和时钟下降沿触发的D触发器。为了弄清这三种器件在功能上的区别，本实验题首先要求对此电路建模，然后用下面给出的测试平台对设计进行仿真测试，将得到的波形图截图后粘贴在下面，对照波形图分析三种器件的功能。

(1) 时钟下降沿触发的D触发器建模：

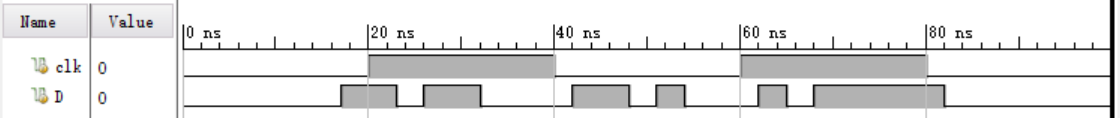
module D\_ff\_n(input clk, input D, output reg Q);

always @ (negedge clk) // 时钟下降沿触发

Q <= D;

endmodule

(2) 测试平台：



`timescale 1ns / 1ps

module lab3\_1\_tb( );

reg clk;

reg D;

wire Qa, Qb, Qc;

initial begin

clk = 1'b0;

#100 $stop;

end

always

#20 clk = !clk;

initial begin

D = 1'b0;

#17 D = !D;

#6 D = !D;

#3 D = !D;

#6 D = !D;

#10 D = !D;

#6 D = !D;

#3 D = !D;

#3 D = !D;

#8 D = !D;

#3 D = !D;

#3 D = !D;

#14 D = !D;

end

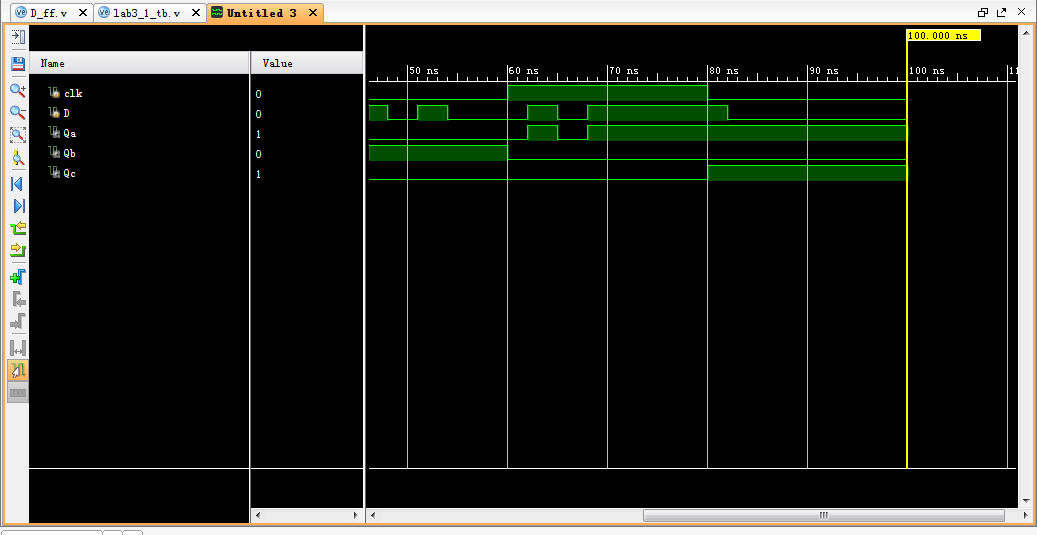
D\_latch myDlatch(clk, D, Qa);

D\_ff myDff(clk, D, Qb);

D\_ff\_n myDffn(clk, D, Qc);

endmodule

(3) 仿真波形图：



2、将几个触发器组合在一起并使用公共时钟，以此保存相关信息，这样的电路称为寄存器。以下是一个带同步复位功能的4bit寄存器。

module Register\_synch\_reset(input [3:0] D, input clk, input rst, output reg [3:0] Q);

always @(posedge clk)

if (rst) begin // 同步复位

Q <= 4'b0;

end else begin

Q <= D;

end

endmodule

下面首先需要在此基础上设计一个带同步复位和使能功能的4bit寄存器，复位信号的优先级要高于使能信号。非复位状态下，该器件在使能信号为高电平时，将输入信号D加载到输出端口Q；否则，输出端口Q不变化。接着，设计测试平台对该寄存器进行仿真测试，观察并分析仿真波形图，验证其功能。

(1) 设计一个带同步复位和使能功能的4bit寄存器：

module Register\_synch\_reset(input [3:0] D, input clk, input rst, input en, output reg [3:0] Q);

always @(posedge clk)

if (rst) begin // 同步复位

Q <= 4'b0;

end

else begin

if (en) begin

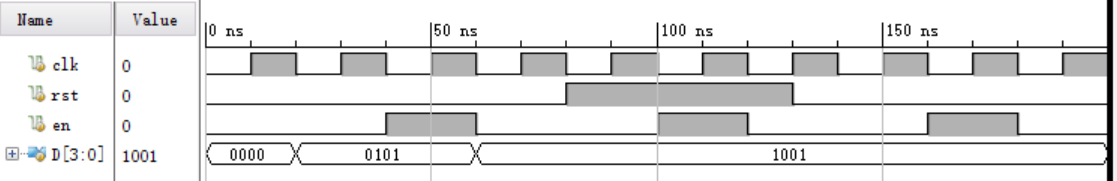
Q<=D;

end

end

endmodule

(2) 设计测试平台进行仿真测试，输入信号波形如下图：



`timescale 1ns / 1ps

module lab3\_2\_tb( );

reg in\_clk,in\_rst,in\_en;

reg [3:0] in\_D;

wire [3:0] out\_Q;

Register\_synch\_reset test(.D(in\_D), .clk(in\_clk), .rst(in\_rst), .en(in\_en), .Q(out\_Q));

initial begin

in\_clk=1'b0;

end

always

#10 in\_clk= !in\_clk;

initial begin

in\_rst=1'b0;

#80 in\_rst= !in\_rst;

#50 in\_rst= !in\_rst;

#70 $stop;

end

initial begin

in\_en=1'b0;

#40 in\_en= !in\_en;

#20 in\_en= !in\_en;

#40 in\_en= !in\_en;

#20 in\_en= !in\_en;

#40 in\_en= !in\_en;

#20 in\_en= !in\_en;

#20 $stop;

end

initial begin

in\_D=4'b0000;

#20 in\_D=4'b0101;

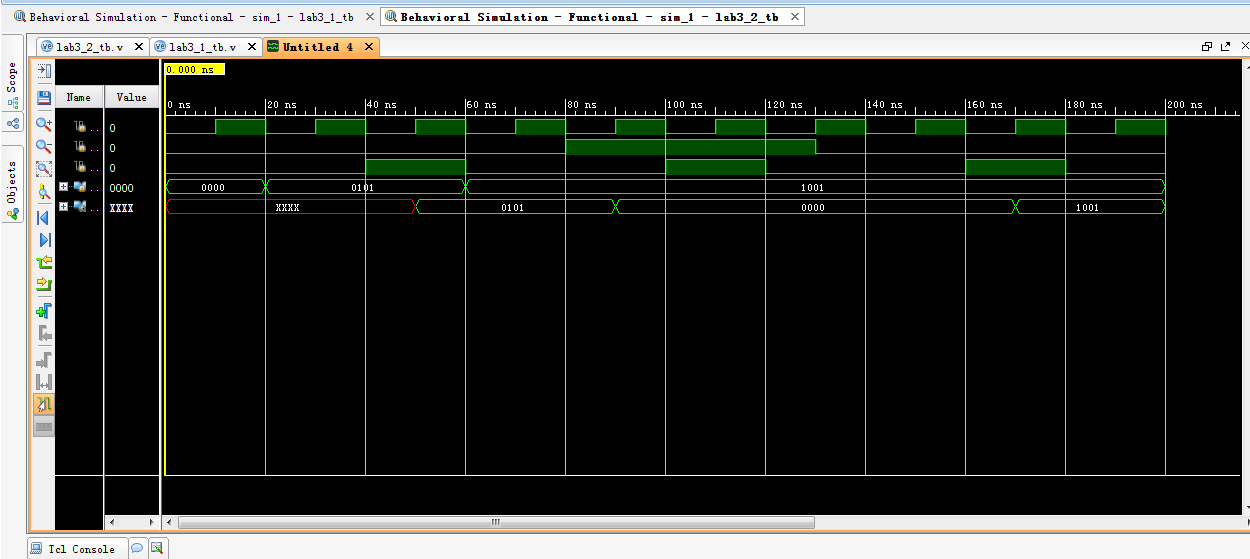
#40 in\_D=4'b1001;

#140 $stop;

end

endmodule

(3) 仿真波形图：



3、下面的代码模拟了一个带加载和移位使能信号的4bit并行输入左移寄存器。

module Parallel\_in\_serial\_out\_load\_enable(clk, ShiftIn, ParallelIn, load, ShiftEn, ShiftOut, RegContent);

input clk, ShiftIn, load, ShiftEn;

input [3:0] ParallelIn;

output ShiftOut;

output [3:0] RegContent;

reg [3:0] shift\_reg;

always @(posedge clk)

if (load)

shift\_reg <= ParallelIn;

else if (ShiftEn)

shift\_reg <= {shift\_reg[2:0], ShiftIn};

assign ShiftOut = shift\_reg[3];

assign RegContent = shift\_reg;

endmodule

下面设计一个4bit串入并出移位寄存器，并用测试平台仿真，输出仿真波形图验证其功能。

(1) 对4bit串入并出移位寄存器建模：

module Serial\_in\_Parallel\_out\_enable(input clk,input ShiftEn,input ShiftIn,output [3:0] ParallelOut,output reg ShiftOut);

reg [3:0] shift\_reg = 4'b0000;

always @(posedge clk)

if (ShiftEn) begin

ShiftOut = shift\_reg[3];

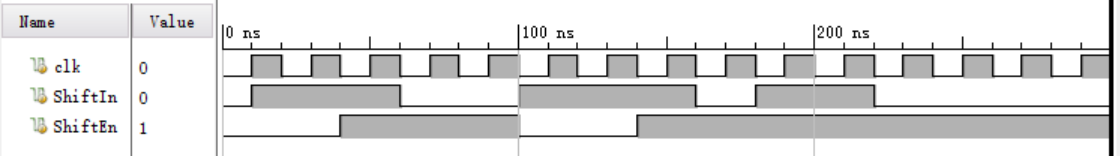
shift\_reg <= {shift\_reg[2:0], ShiftIn};

end

assign ParallelOut = shift\_reg;

endmodule

(2) 设计测试平台进行仿真测试，输入信号波形如下图：



`timescale 1ns / 1ps

module lab3\_3\_tb();

reg in\_clk,in\_ShiftEn,in\_ShiftIn;

wire [3:0] ParallelOut\_out;

wire ShiftOut\_out;

Serial\_in\_Parallel\_out\_enable test(.clk(in\_clk),.ShiftEn(in\_ShiftEn),.ShiftIn(in\_ShiftIn),.ShiftOut(ShiftOut\_out),.ParallelOut(ParallelOut\_out));

initial begin

in\_clk=1'b0;

end

always

#10 in\_clk= !in\_clk;

initial begin

in\_ShiftIn=1'b0;

#10 in\_ShiftIn= !in\_ShiftIn;

#50 in\_ShiftIn= !in\_ShiftIn;

#40 in\_ShiftIn= !in\_ShiftIn;

#60 in\_ShiftIn= !in\_ShiftIn;

#20 in\_ShiftIn= !in\_ShiftIn;

#40 in\_ShiftIn= !in\_ShiftIn;

#80 $stop;

end

initial begin

in\_ShiftEn=1'b0;

#40 in\_ShiftEn= !in\_ShiftEn;

#60 in\_ShiftEn= !in\_ShiftEn;

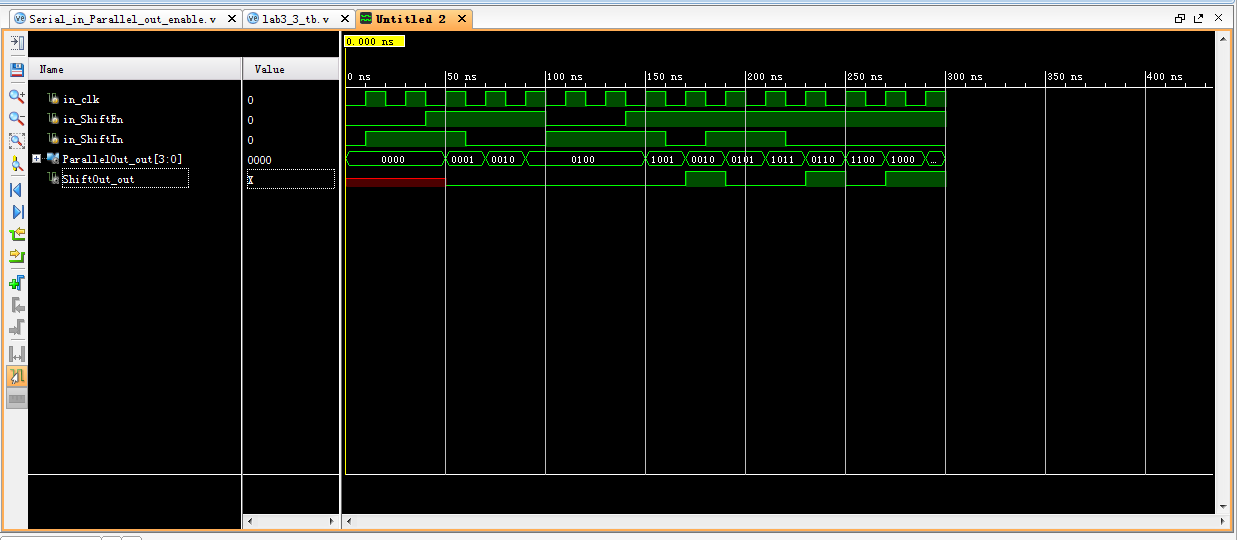
#40 in\_ShiftEn= !in\_ShiftEn;

#160 $stop;

end

endmodule

(3) 仿真波形图：



4、下面给出的是一个简单的4bit加法计数器：

module CNT4(input clk, outut [3:0] Q);

reg [3:0] Q1;

always @(posedge clk)

Q1 <= Q1 + 1;

assign Q = Q1;

endmodule

(1) 设计一个带同步复位和使能功能的4bit加法计算器（复位优先级高于使能优先级）：

module CNT4\_synch\_reset\_enable(input clk, input rst, input en, output reg [3:0] Q);

always @(posedge clk)

if (rst) begin // 同步复位

Q <= 4'b0;

end

else begin

if (en) begin

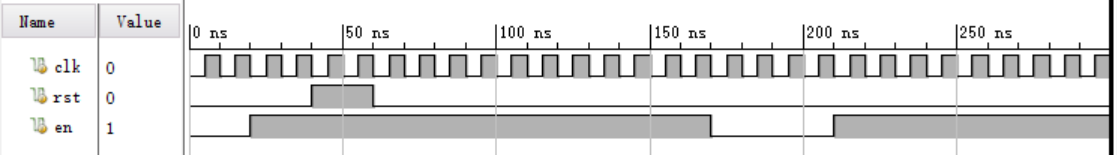
Q<=Q+1;

end

end

endmodule

(2) 设计测试平台进行仿真测试，输入信号波形如下图：



`timescale 1ns / 1ps

module lab3\_4\_tb();

reg clk;

reg en,rst;

wire [3:0] Q;

initial begin

clk = 1'b0;

end

always

#5 clk = !clk;

initial begin

rst = 1'b0;

#40 rst = !rst;

#20 rst = !rst;

#240 $stop;

end

initial begin

en = 1'b0;

#20 en = !en;

#150 en = !en;

#40 en = !en;

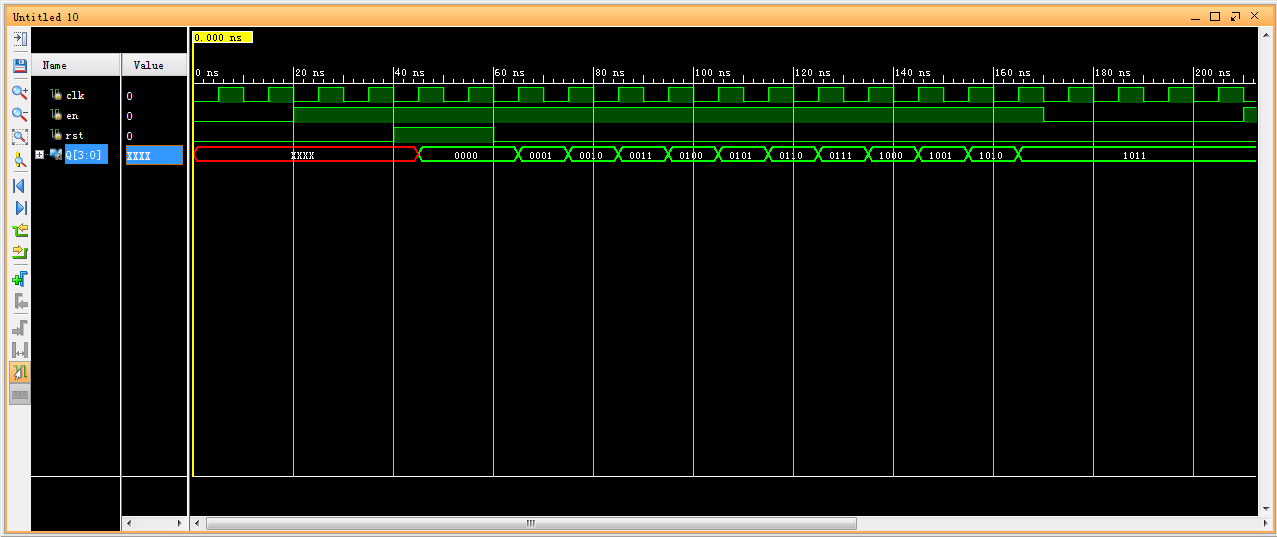
#90 $stop;

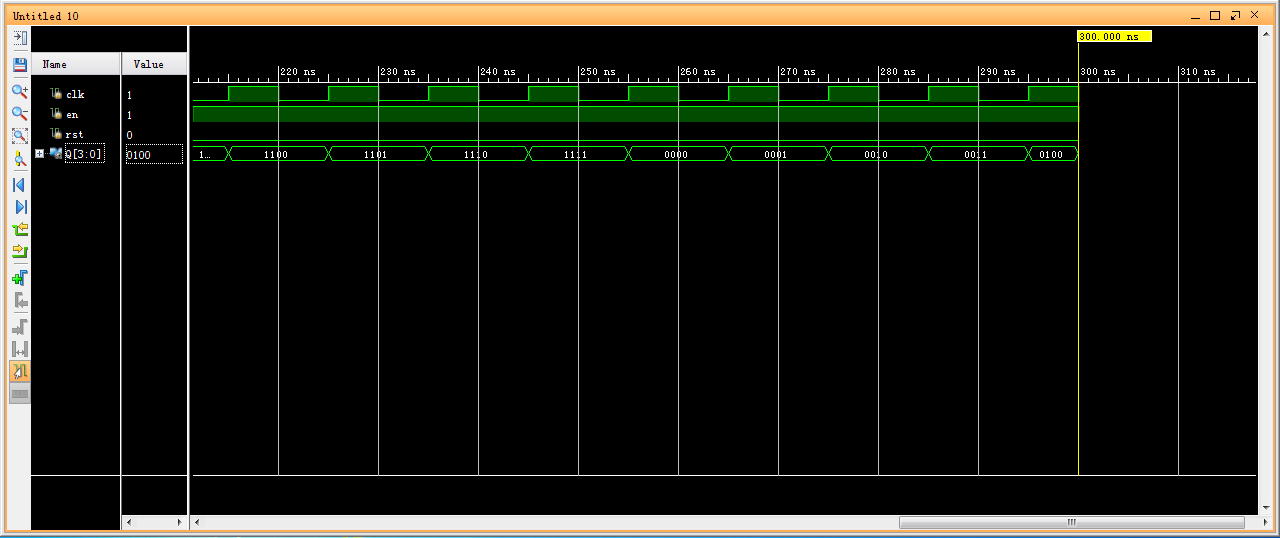
end

CNT4\_synch\_reset\_enable test(clk, rst, en, Q);

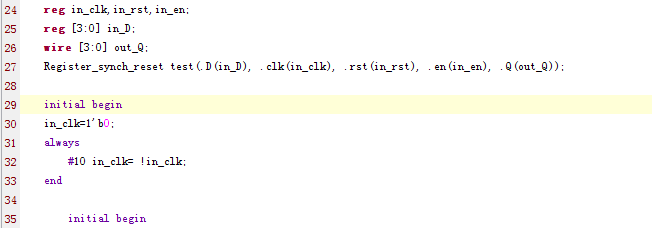
Endmodule

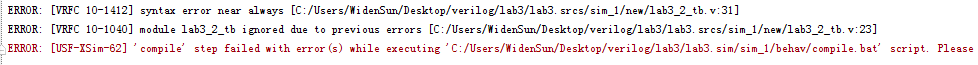
(3) 仿真波形图：





## 遇到的问题及解决方法





这次试验在编写第一个原件电路是比较顺利，但是在编写测试文件的时候，有一个报错我看了好久也没明白怎么回事，一直说我的always语句有问题，开始我以为是漏了end或是begin，最后才发现，原来always语句不能在嵌套中使用，果然，修改过后，报错消失了。

编写第二个文件和测试比较顺利。但是在第三个任务的时候我卡住了，怎么都没想明白这个任务是让我怎么做，后来我反复观看了老师给我的教学视频后，才渐渐理解了如何做到串行输入，并行输出。

这次实验在编写测试文件的时候我有个疑问，在always语句中，如果我只让程序循环10次，应该怎样让它停下来呢，最后我只想到了再多申请一个寄存器变量用来作为计数器，判断语句就写成计数器是否清0就可以了，当然个人认为还是比较浪费空间的，希望能够像汇编语言一样专门分配寄存器用来做循环的计数器就好了。

## 实验心得、意见和建议

这次实验课，总的来说自己已经上路了，可以独立的写出自己需要的电路并编写测试文件。当然这次我遇到了应该如何在自己希望的地方终止always，目前的解决办法是换成循环再加个计数器判定，此外感觉对于仿真波形图，vivado只能一级一级放大缩小，没有像word那样可以按照百分比放缩，我经常不能让图像放大到自己最满意的状态，希望程序能够改进的更加人性化。