# lab6

实验内容:掌握触发器的描述 并实现寄存器,递减器来加深了解

PB17030889 吉志远

### 各个实验总结:

lab6_1_1 没啥好说的就是简单的D触发器组实现一个寄存器,在clk上升沿更新一下

lab6_2_2 这个比较坑要是跳过了lab6_2_1 的铺垫偷懒只做2_2会一脸懵逼导致无所适从,结果发现只要按照2_1给的电路并用Dff实现Tf即可f

lab6_2_3

//记录一次debug

lab6的pdf上写的是cnt_done=~|count

明显是错误的根据逻辑判断此处应为判断count是否4位都为0所以第一次是

> ~count[0]|count[1]|count[2]|count[3]

结果出错, 故 debug: print大法

结果发现cnt_done几乎全都为1,采用了这里代码的第二种方法

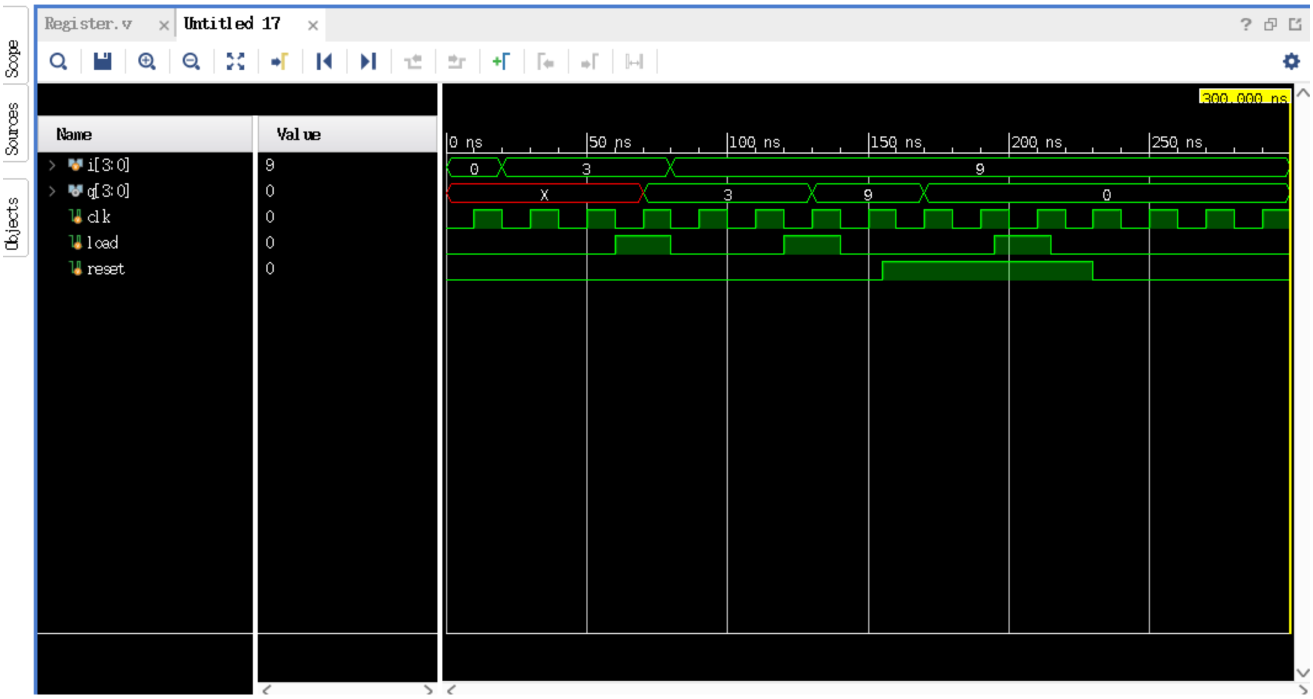
## lab6_1_1

### lab6_1_1 代码

```
`timescale 1ns / 1ps
```

```
module Register_with_synch_reset_load_behavior(
    input [3:0] D,
    input clk,
    input reset,
    input load,
    output reg [3:0]Q
    );
    always @(posedge clk)
        if(reset)
         begin
            Q <= 4'b0;
         end
         else if (load) begin
            Q<=D;
    end
endmodule
```

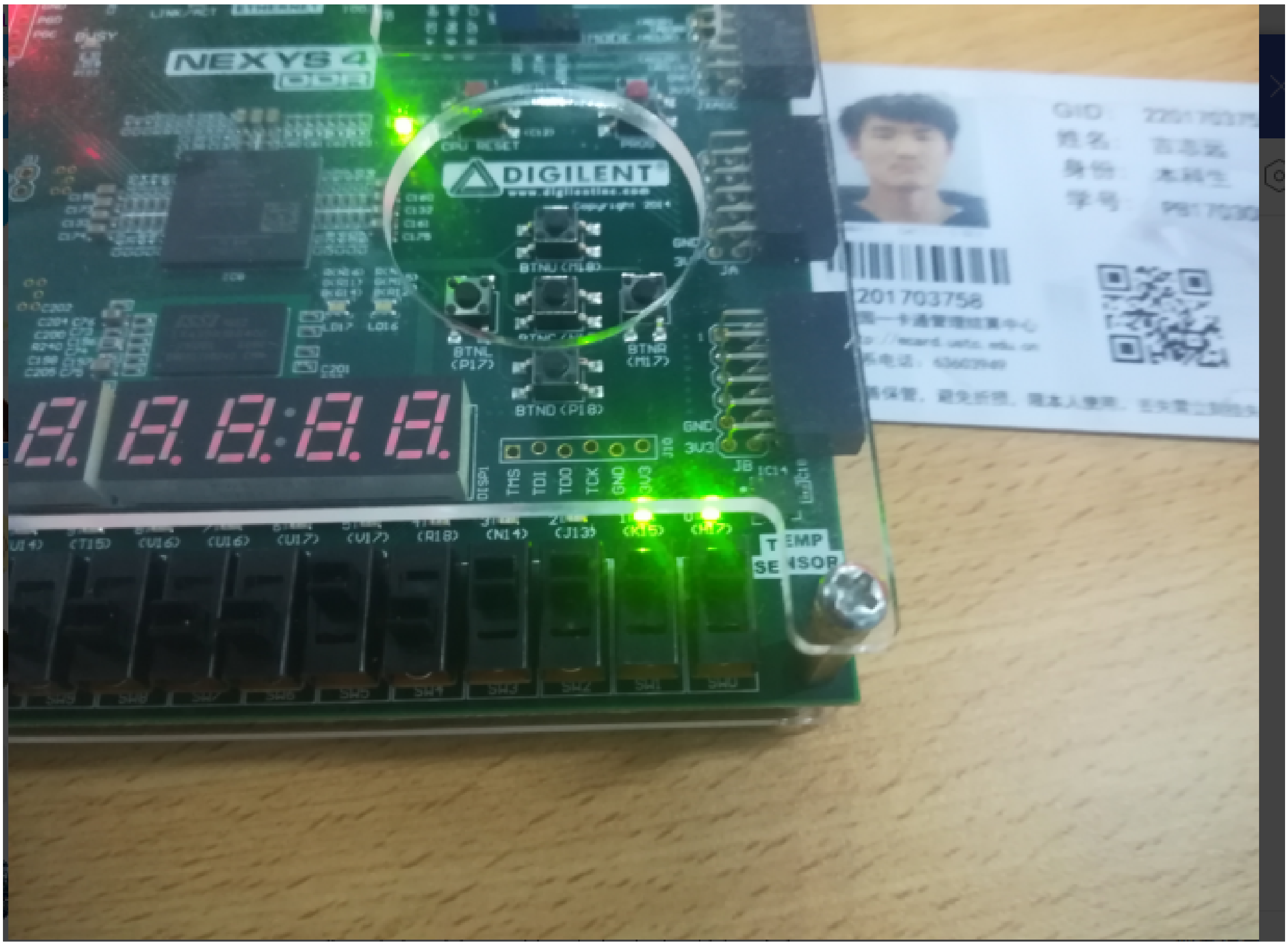
## lab6_1_1 仿真截图



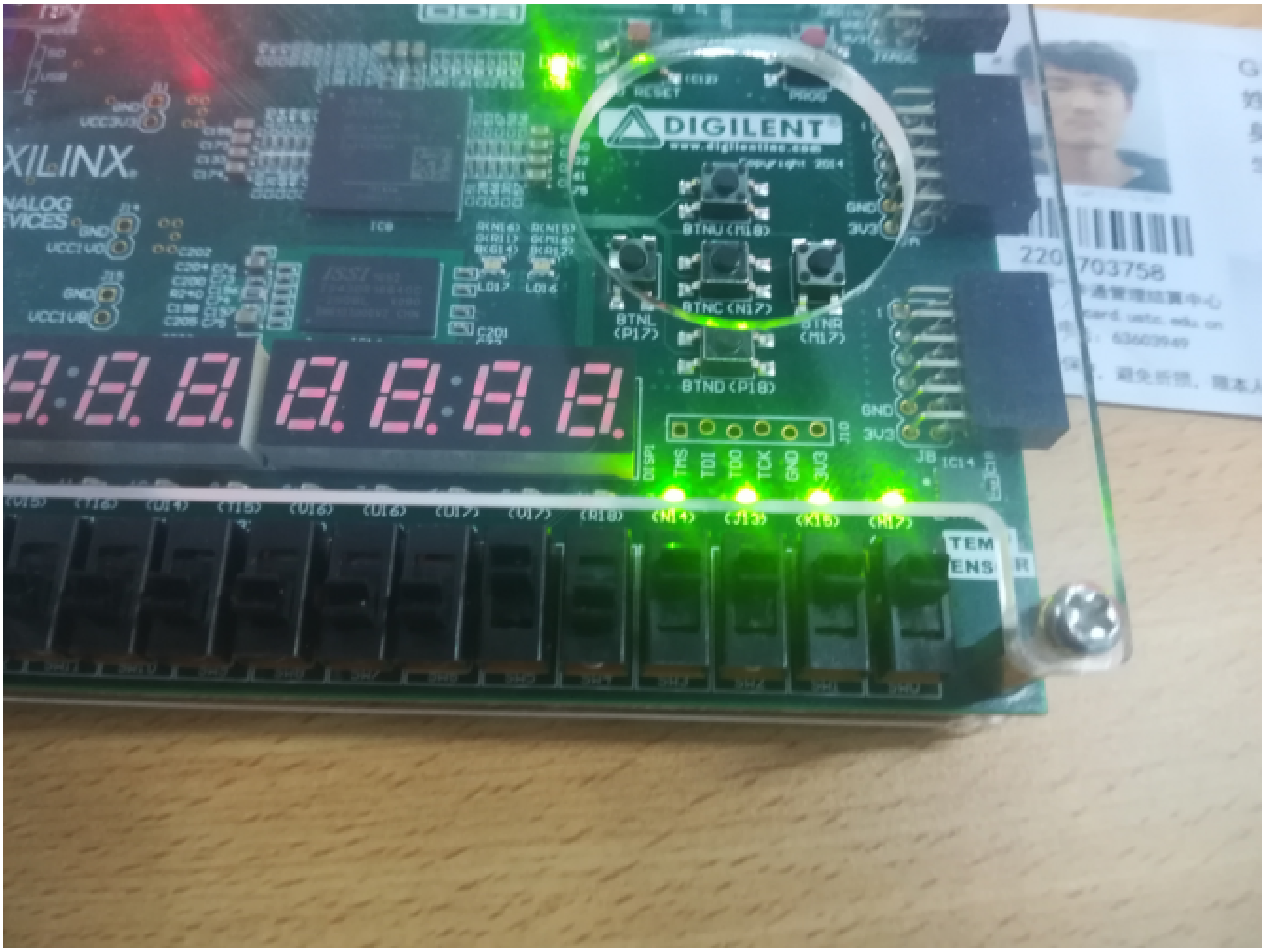
| Name | Value |
|---|---|
| i[3:0] | 9 |
| q[3:0] | 0 |
| clk | 0 |
| load | 0 |
| reset | 0 |

## lab6_1_1 bitstream

(递增swt15前)



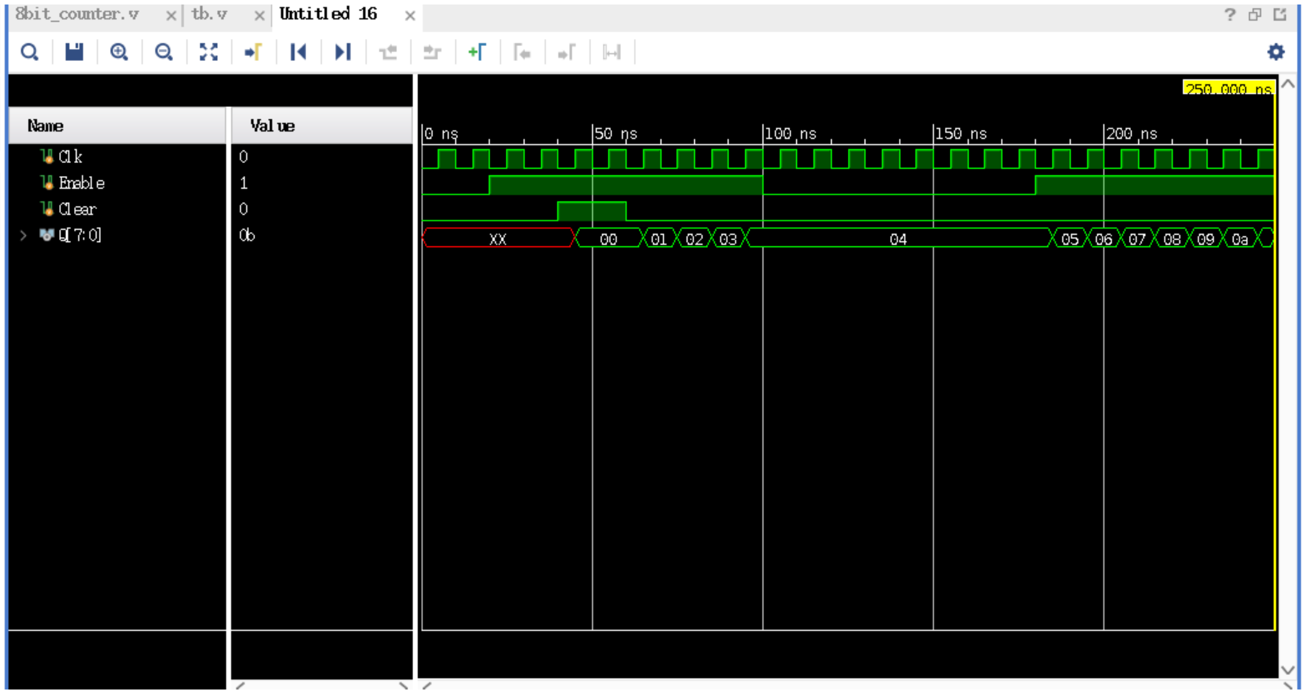
(递增后)



# lab6_2_2

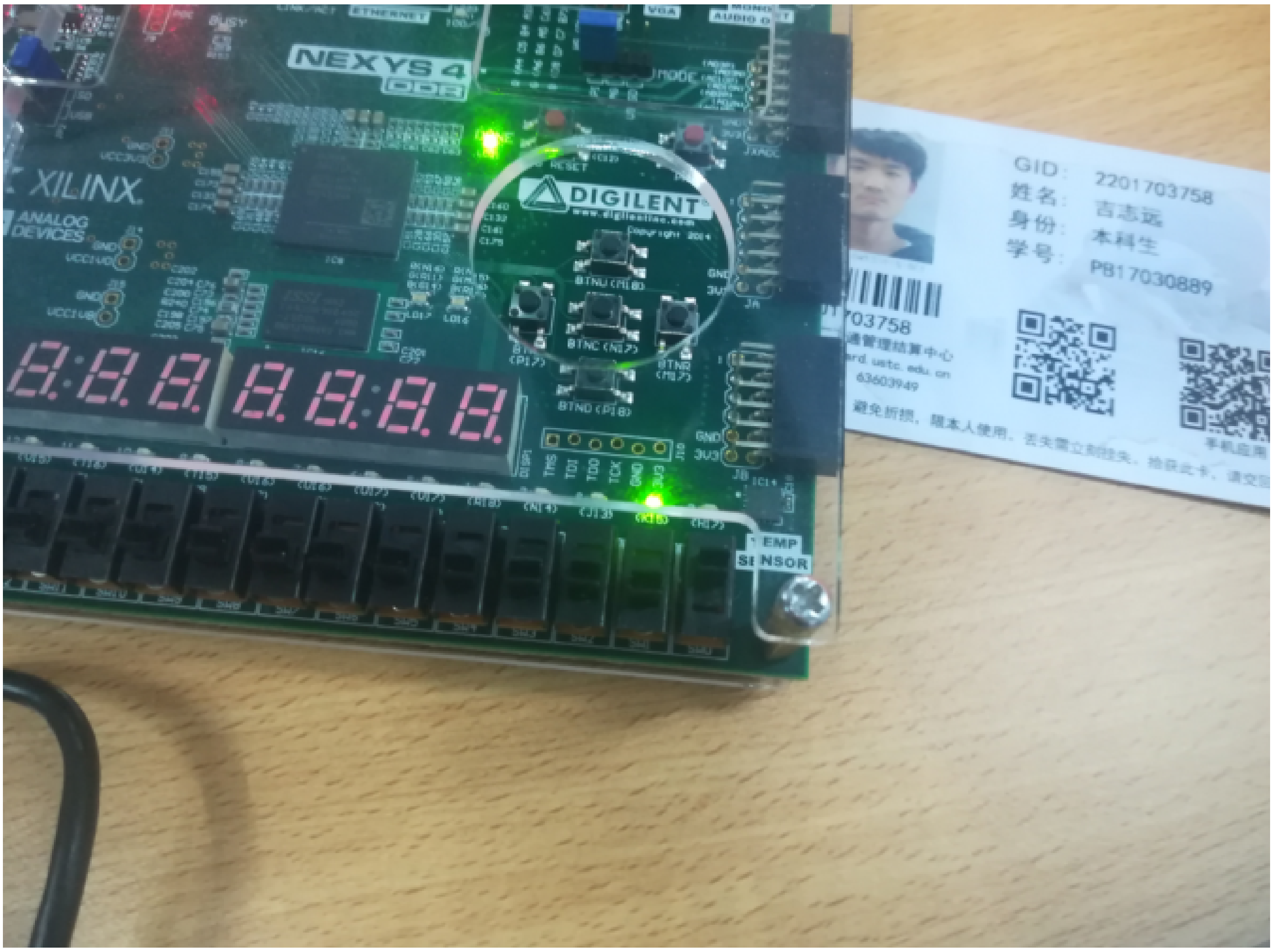
## lab6_2_2 代码

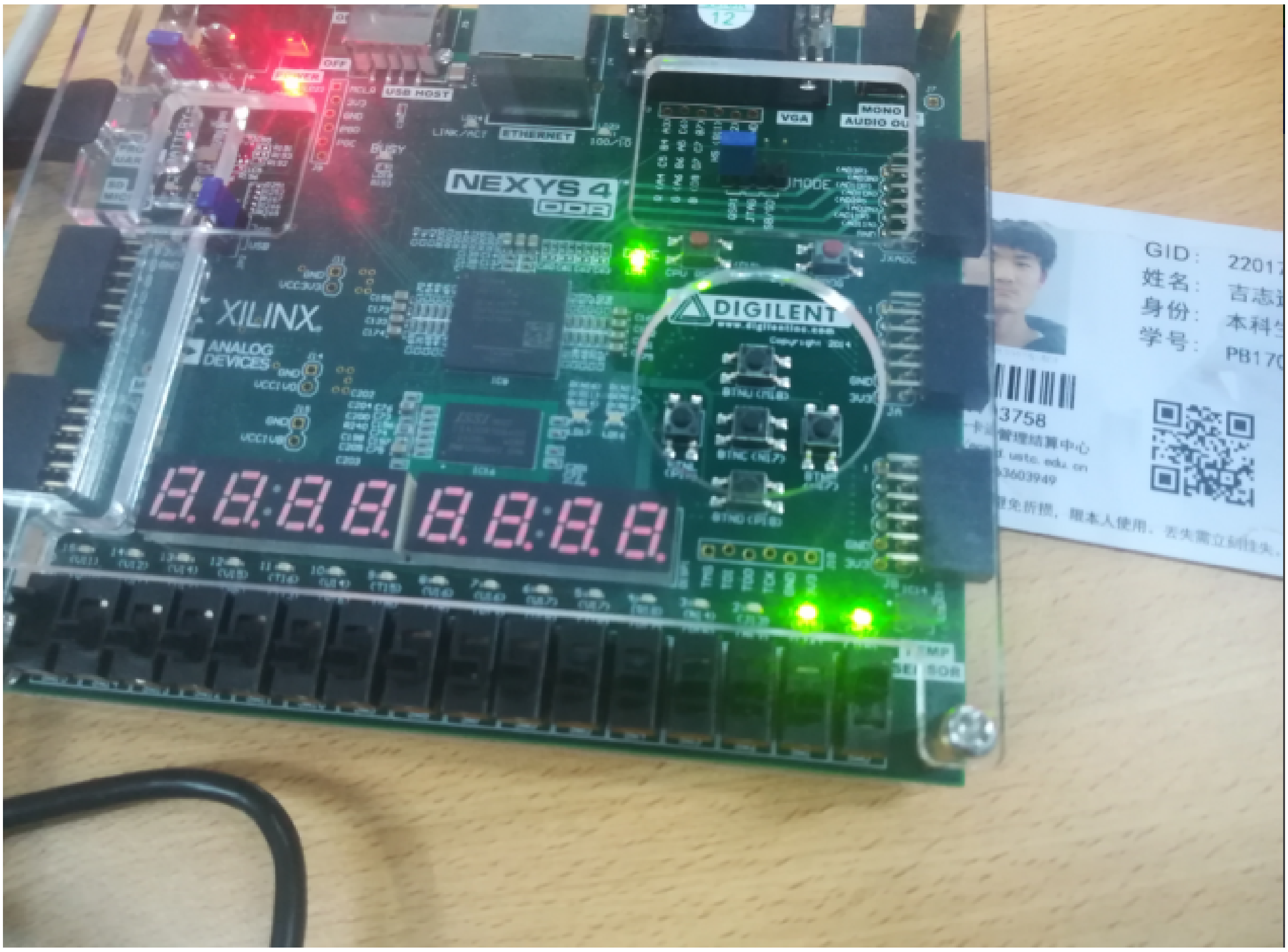
```
`timescale 1ns / 1ps

module eightbit_counter(
    output [7:0]Q,
    input Enable,
    input Clk,
    input Clear
    );
    wire a,b,c,d,e,f,h;
    T_FF FF0 (Enable,Clk,Clear,Q[0]);
    assign a = Enable&Q[0];
    T_FF FF1 (a,Clk,Clear,Q[1]);
    assign b = a&Q[1];
    T_FF FF2 (b,Clk,Clear,Q[2]);
    assign c = b&Q[2];
    T_FF FF3 (c,Clk,Clear,Q[3]);
    assign d = c&Q[3];
    T_FF FF4 (d,Clk,Clear,Q[4]);
    assign e = d&Q[4];
    T_FF FF5 (e,Clk,Clear,Q[5]);
    assign f = e&Q[5];
    T_FF FF6 (f,Clk,Clear,Q[6]);
    assign h = f&Q[6];
    T_FF FF7 (h,Clk,Clear,Q[7]);
endmodule
module D_FF(input D, input Clk, output reg Q);
    always @ (posedge Clk)
        Q <= D;

endmodule

module T_FF(input T,input Clk,input clear, output reg Q);
    wire a;
    assign a = ~Q&T+T&~Q;
    D_FF FF (a,Clk,Q);
endmodule
// T 触发器的描述
/*module T_FF (input T,input Clk,input clear,output reg Q);
    always @ (posedge Clk)
        begin
        if(clear)
        Q <= 1'b0;
        else if (T)
        Q <= ~Q;
        else
        Q <=Q;
        end
endmodule*/
```



| Name | Value |
|---|---|
| Clk | 0 |
| Enable | 1 |
| Clear | 0 |
| Q[7:0] | 0b |

## lab6_2_2 bitstream


NEXYS 4
DDR
XILINX
ANALOG
DEVICES
DIGILENT
BTNU (M18)
BTNC (N17)
BTNR (M17)
BTND (P18)
RESET
TEMP
SENSOR
GID:
2201703758
姓名:
吉志远
身份:
本科生
学号:
PB17030889
03758
手机应用



(用swt[15]模拟Clk 来实现递增)

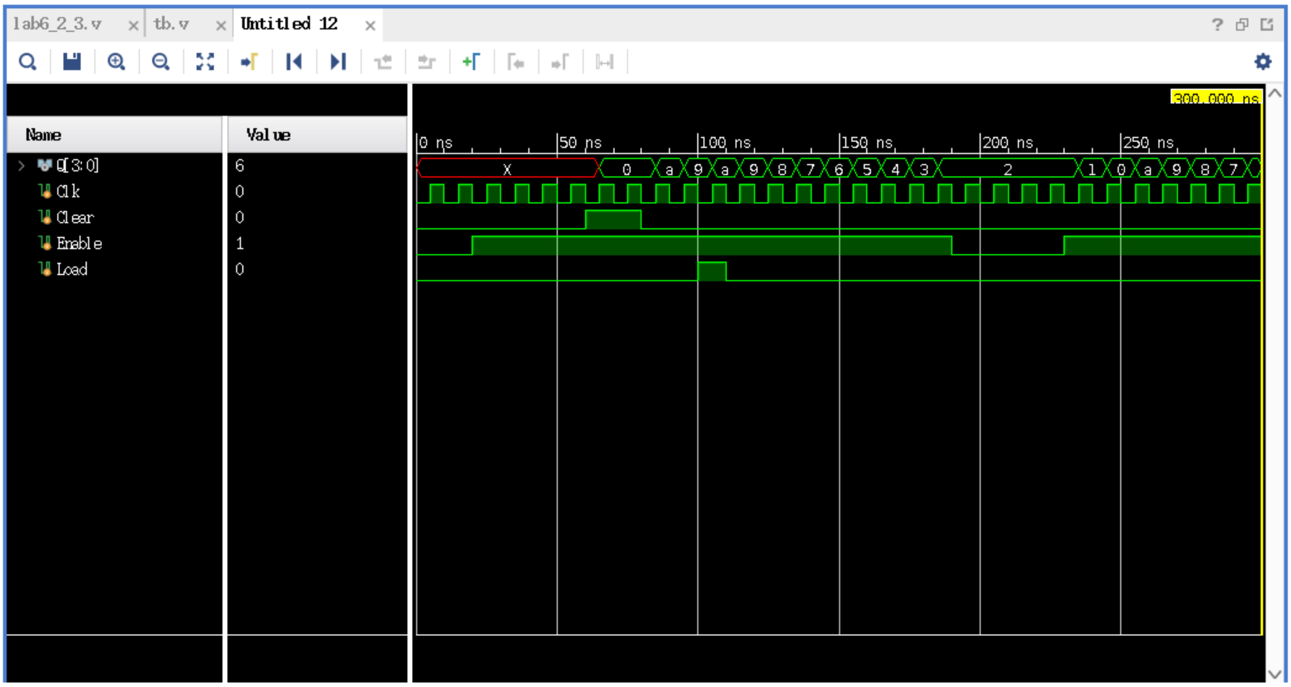
# lab6_2_3

## lab6_2_3 代码

```
`timescale 1ns / 1ps

module lab6_2_3(
    output reg[3:0] count,
    //input cnt_done,
    input Clock,
    input Clear,
    input Enable,
    input Load
    );
    wire Q;
    wire cnt_done;
    assign cnt_done = (count==4'b0000)?1:0;
    assign Q = count;
    //assign swith = Load|cnt_done;
    always @ (posedge Clock)
        if(Clear)
            count <= 0;
        else if (Enable)
            if(Load | cnt_done)
```

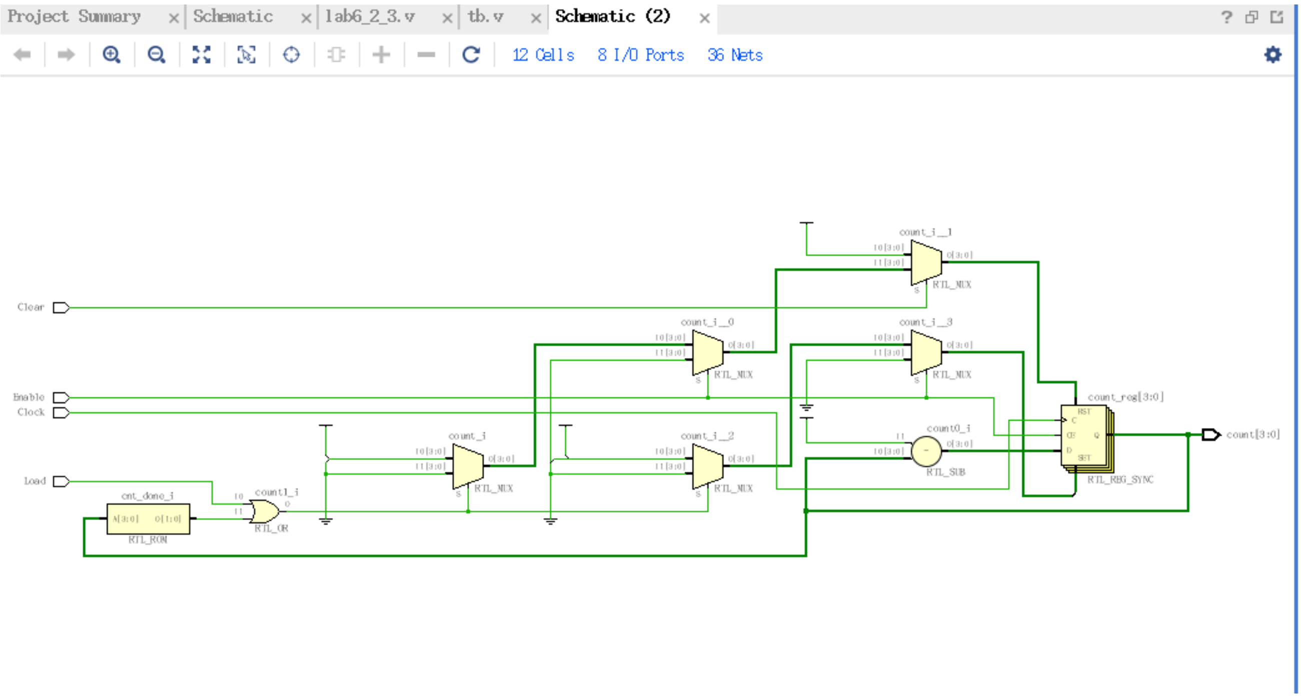
```
                count <= 4'd10;
            else
            count <= count - 1;
    endmodule
```

## lab6_2_3 仿真



```
Name        Value
Q[3:0]      6
Clk         0
Clear       0
Enable      1
Load        0
```

## lab6_2_3 原理图



Project Summary
Schematic
lab6_2_3.v
tb.v
Schematic (2)
12 Cells
8 I/O Ports
36 Nets
Clear
Enable
Clock
Load
cnt_done_i
RTL_ROM
count1_i
RTL_OR
count_i
RTL_MUX
count_i__0
RTL_MUX
count_i__1
RTL_MUX
count_i__2
RTL_MUX
count_i__3
RTL_MUX
count0_i
RTL_SUB
count_reg[3:0]
RTL_REG_SYNC
count[3:0]