实验报告

实验题目: 计数器，计时器 和 实时时钟

日期: 2018 年 11 月 23 日

姓名: 罗晏宸 学号: PB17000297 成绩:

实验目的:

1. 学习如何定义参数化模型，掌握建立更泛用模块的编写风格
2. 使用行为建模与 IP 目录的例化分别设计计数器，并加以对比
3. 使用内核和用 HDL 建模的附加电路设计定时器电路
4. 通过掌握的知识，创建实时的时钟

实验内容（截图、照片与代码）

Lab9_1_1

代码 lab9_1_1.v

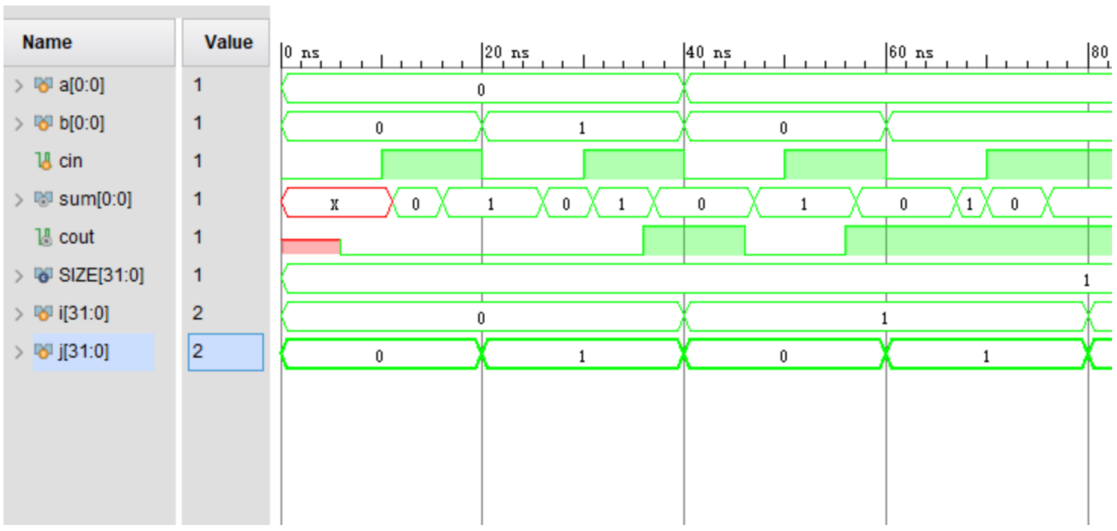
```
`timescale 1ns / 1ps
module lab9_1_1(a,b,cin,sum,cout);
    parameter SIZE = 1;
    input [SIZE-1:0] a,b;
    output [SIZE-1:0] sum;
    input cin;
    output cout;
    wire [SIZE-1:1] temp;

    defparam uFULLADDER.NOT_DELAY=1, uFULLADDER.AND_DELAY=3,
uFULLADDER.OR_DELAY=3;

    fulladder_gatelevel
        uFULLADDER (.a(a[0]),.b(b[0]),.cin(cin),.sum(sum),.cout(cout));

endmodule
```

代码 fulladder_gatelevel.v

```
`timescale 1ns / 1ps
module fulladder_gatelevel (a,b,cin,sum,cout);
    parameter AND_DELAY=2, OR_DELAY=2, NOT_DELAY=2;
    input a,b,cin;
    output sum,cout;
    wire q1,q2,q3,r1,r2,s,t1,t2,t3,_a,_b,_cin,_r1,_r2;
    and #(AND_DELAY)
        uAND1 (q1,a,b),
        uAND2 (q2,b,cin),
        uAND3 (q3,a,cin),
        uAND4 (s,r1,r2),
        uAND5 (sum,t1,t2,t3);
    or #(OR_DELAY)
        uOR1 (cout,q1,q2,q3),
        uOR2 (r1,_a,b),
        uOR3 (r2,_b,a),
        uOR4 (t1,_r1,_r2,cin),
        uOR5 (t2,_cin,r1),
        uOR6 (t3,_cin,r2);
    not #(NOT_DELAY)
        uNOT1 (_a,a),
        uNOT2 (_b,b),
        uNOT3 (_cin,cin),
        uNOT4 (_r1,r1),
        uNOT5 (_r2,r2);

endmodule
```

代码 lab9_1_1_tb.v

```
`timescale 1ns / 1ps
module lab9_1_1_tb();
    parameter SIZE = 1;
    reg [SIZE-1:0] a,b;
    reg cin;
    integer i,j;
    wire cout;
    wire [SIZE-1:0] sum;

    lab9_1_1 DUT (.a(a), .b(b), .cin(cin), .sum(sum), .cout(cout));

    initial
    begin
        cin = 0;
        for ( i = 0 ; i < 1<<SIZE ; i = i + 1 )
        begin
            a = i;
            for ( j = 0 ; j < 1<<SIZE ; j = j + 1 )
            begin
                b = j;
                cin = 0;#10
                cin = 1;#10;
            end
        end
    end
endmodule
```



截图 1-lab9_1_1 仿真

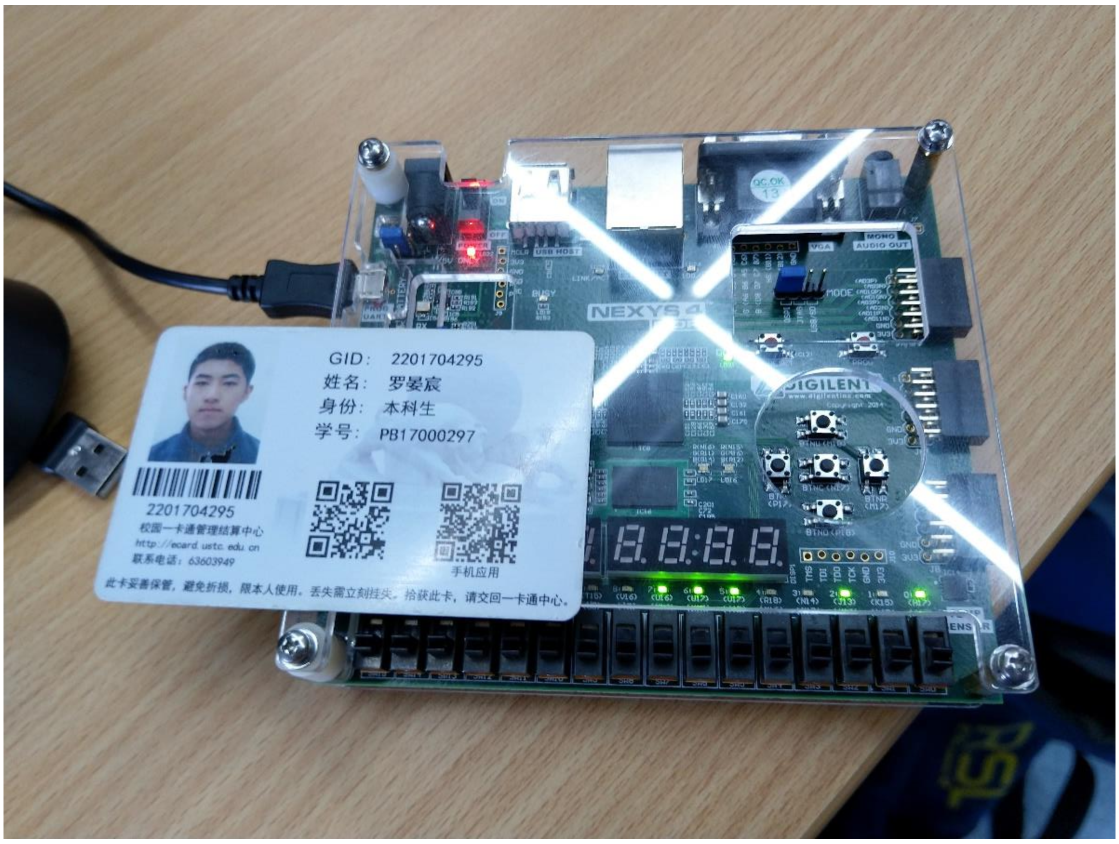
lab9_2_1

代码 up_down_Counter.v

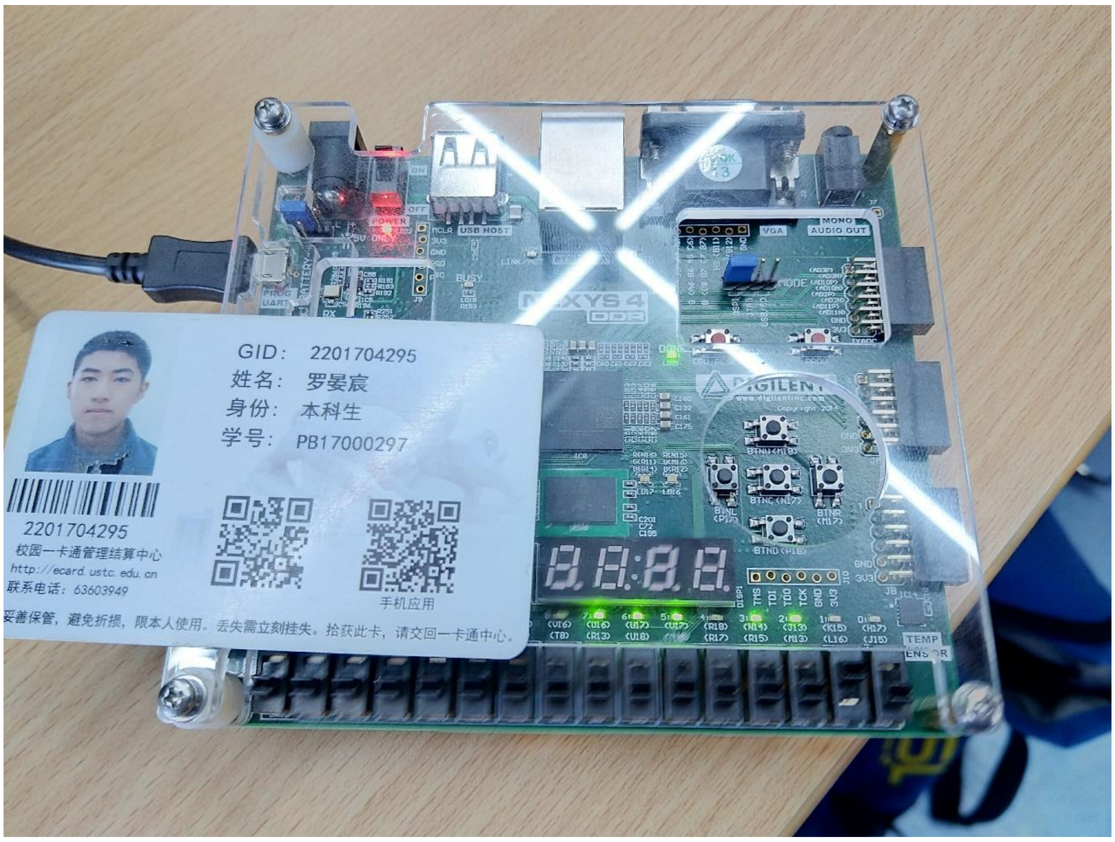
```
`timescale 1ns / 1ps
 (* use_dsp48 = "no" *)
module up_down_Counter #(parameter COUNT_SIZE = 8) (
    input CLK100MHZ,
    input up_down,
    input enable,
    input reset,
    output reg [COUNT_SIZE - 1:0] Count
    );

    wire pulse,clk;
    clk_wiz_0 CLOCK5M (clk,CLK100MHZ);
    Counter CLOCK (clk,~enable,pulse);
    always@(posedge pulse)
    begin
        if (up_down)
        begin
            if ((Count >= (1 << COUNT_SIZE) - 1) || (reset))
                Count   <= 'b0;
            else
                Count   <= Count + 'b1;
        end
        else
        begin
            if ((Count == 'b0) || (reset))
                Count   <= (1 << COUNT_SIZE) - 1;
            else
                Count   <= Count - 'b1;
        end
    end

endmodule
```

代码 Counter.v

```
`timescale 1ns / 1ps
module Counter(input clk,input rst,output pulse);
     reg [23:0] cnt = 0;
     always @(posedge clk or posedge rst)
     begin
        if (rst)
        cnt <= 23'd0;
        else
            if(cnt >= 23'd499999)
            cnt <= 23'd0;
        else
            cnt <= cnt + 23'd1;
     end
    assign pulse =  (cnt == 23'd499999) ? 1'b1 : 1'b0;
endmodule
```



照片 1-lab9_2_1 下载 A



照片 2-lab9_2_1 下载 B

“照片 1-lab9_2_1 下载 A”与“照片 2-lab9_2_1 下载 B”的说明：本工程使用行为建模设计一个 8 位向上/向下计数器，A/B 中 switch[1]分别为开与关，表明计数器正处于向上/向下的计数状态，led[0]~led[7]的亮灭表示 count 的物理显示，图 A/B 中 count 分别为$(11100101)_B$和$(11101100)_B$，表示彼时计数为$(229)_D$与$(236)_D$。

Lab9_3_1

代码 lab9_3_1.v

```
`timescale 1ns / 1ps
module lab9_3_1(input CLK100MHZ,input reset,input enable,output [7:0]
AN,output [7:0] seg);
    wire tick,second,clk;
    wire [1:0] pulse;
    wire [2:0] threshold;
    wire [3:0] Q0,Q1,Q2,Q3;
    wire [7:0] seg0,seg1,seg2,seg3;
    clk_wiz_0(clk,CLK100MHZ);
    Counter0(clk,~enable,tick);
    c_counter_binary_0 CLOCK1 (tick,1,reset,threshold[0],Q0);
    c_counter_binary_0 CLOCK2 (tick,threshold[0] & Q0[3] &
Q0[0],reset,threshold[1],Q1);
    c_counter_binary_1 CLOCK3 (tick,threshold[1] &
threshold[0],reset,threshold[2],Q2);
    c_counter_binary_2 CLOCK4 (tick,threshold[2] & threshold[1] &
threshold[0],reset,Q3);
    Counter2(clk,pulse);
    assign seg0[7] = 1;
    assign seg1[7] = 0;
    assign seg2[7] = 1;
    assign seg3[7] = 0;
    bcdto7segment_dataflow (Q0,seg0[6:0]);
    bcdto7segment_dataflow (Q1,seg1[6:0]);
    bcdto7segment_dataflow (Q2,seg2[6:0]);
    bcdto7segment_dataflow (Q3,seg3[6:0]);
    assign seg = (pulse<=2'b01) ? ((pulse==2'b00) ? seg0 : seg1) :
((pulse==2'b10) ? seg2  : seg3);
    assign AN[0] = (pulse==2'b00) ? 0 : 1;
    assign AN[1] = (pulse==2'b01) ? 0 : 1;
    assign AN[2] = (pulse==2'b10) ? 0 : 1;
    assign AN[3] = (pulse==2'b11) ? 0 : 1;
    assign AN[4] = 1;
    assign AN[5] = 1;
    assign AN[6] = 1;
    assign AN[7] = 1;
endmodule
```

代码 Counter.v

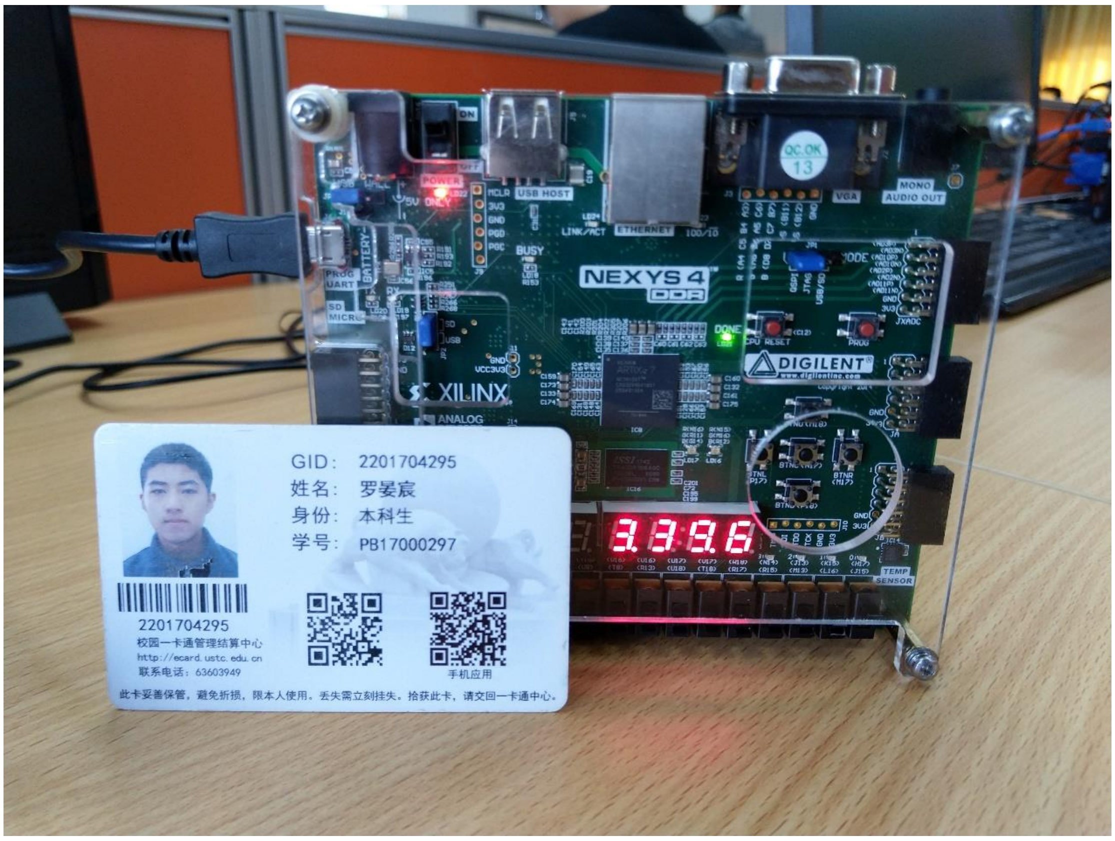
```
`timescale 1ns / 1ps
module Counter0(input clk,input rst,output pulse);
     reg [23:0] cnt = 0;
     always @(posedge clk or posedge rst)
     begin
        if (rst)
        cnt <= 23'd0;
        else
            if(cnt >= 23'd499999)
            cnt <= 23'd0;
        else
            cnt <= cnt + 23'd1;
     end
    assign pulse =  (cnt == 23'd499999) ? 1'b1 : 1'b0;
endmodule
```

代码 Counter2.v

```
`timescale 1ns / 1ps
module Counter2(input clk,output reg [2:0] pulse);
     reg [20:0] cnt;
     always @(posedge clk)
     begin
        if(cnt >= 'd24999)
        begin
            cnt <= 'd0;
            pulse <= pulse + 'b1;
        end
        else
            cnt <= cnt + 'd1;
     end
endmodule
```

代码 bcdto7segment_dataflow.v

```
`timescale 1ns / 1ps
module bcdto7segment_dataflow(
    input [3:0] x,
    output [6:0] seg
    );
    assign seg[0] = ((~x[3] && ~x[2] && ~x[1] && x[0]) | (~x[3] && x[2]
&& ~x[1] && ~x[0])) && ~(x[3] && (x[2] | x[1]));
    assign seg[1] = (~x[3] && x[2] && ~x[1] && x[0]) | (~x[3] && x[2] &&
x[1] && ~x[0]) | (x[3] && (x[2] | x[1]));
    assign seg[2] = (~x[3] && ~x[2] && x[1] && ~x[0]) | (x[3] && (x[2] |
x[1]));
    assign seg[3] = ((~x[3] && ~x[2] && ~x[1] && x[0]) | (~x[3] && x[2]
&& ~x[1] && ~x[0]) | (~x[3] && x[2] && x[1] && x[0])) && ~(x[3] &&
(x[2] | x[1]));
    assign seg[4] = ((~x[3] && ~x[2] && ~x[1] && x[0]) | (~x[3] && ~x[2]
&& x[1] && x[0]) | (~x[3] && x[2] && ~x[1] && ~x[0]) | (~x[3] && x[2]
&& ~x[1] && x[0]) | (~x[3] && x[2] && x[1] && x[0]) | (x[3] && ~x[2] &&
~x[1] && x[0])) && ~(x[3] && (x[2] | x[1]));
    assign seg[5] = ((~x[3] && ~x[2] && ~x[1] && x[0]) | (~x[3] && ~x[2]
&& x[1] && ~x[0]) | (~x[3] && ~x[2] && x[1] && x[0]) | (~x[3] && x[2]
&& x[1] && x[0])) && ~(x[3] && (x[2] | x[1]));
    assign seg[6] = ((~x[3] && ~x[2] && ~x[1] && ~x[0]) | (~x[3] &&
~x[2] && ~x[1] && x[0]) | (~x[3] && x[2] && x[1] && x[0])) && ~(x[3] &&
(x[2] | x[1]));
endmodule
```



照片 3-lab9_3_1 下载

“照片 4-lab9_3_1 下载”的说明：本工程使用数据流建模与例化 IP 目录实现一个精确到 0.1s 的在七段数码管上以 M.SS.f 的格式显示的带有复位和使能功能的计时器。图中右侧四位七段数码管的显示为”3.39.6”字样 表明彼时计时为 3’39.6’’

实验总结

本次实验中，学习了如何实现模型的参数化，便于在后续的学习与工程设计中实现更为泛用的模块；另外，通过对行为建模与例化 IP 对于计数器的不同实现，了解了计数与计时的实现原理，进一步的，利用已学习的知识设计了相对有实际意义的时钟应用，通过实践巩固了以上所设计的开发技术。

课后习题

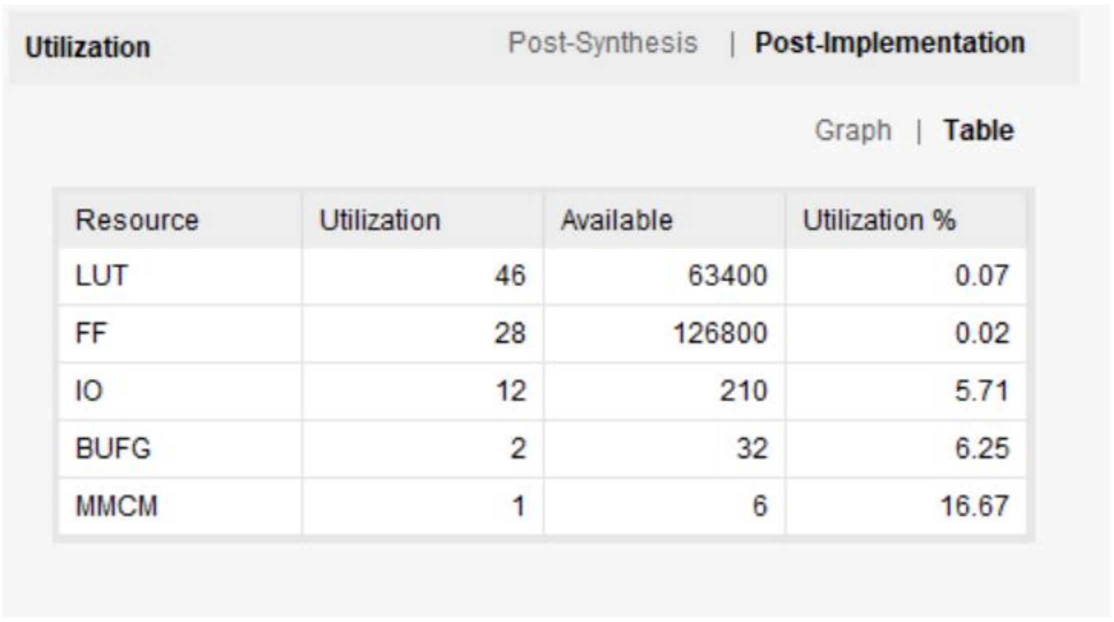
Lab9_2_1

代码 up_down_Counter.v

```
`timescale 1ns / 1ps
 (* use_dsp48 = "no" *)
module up_down_Counter #(parameter COUNT_SIZE = 8) (
    input CLK100MHZ,
    input up_down,
    input enable,
    input reset,
    output reg [COUNT_SIZE - 1:0] Count
    );

    wire pulse,clk;
    clk_wiz_0 CLOCK5M (clk,CLK100MHZ);
     Counter CLOCK (clk,~enable,pulse);
    always@(posedge pulse)
    begin
        if (up_down)
        begin
            if ((Count >= (1 << COUNT_SIZE) - 1) || (reset))
                Count   <= 'b0;
            else
                Count   <= Count + 'b1;
        end
        else
        begin
            if ((Count == 'b0) || (reset))
                Count   <= (1 << COUNT_SIZE) - 1;
            else
                Count   <= Count - 'b1;
        end
    end

endmodule
```

代码 Counter.v

```
`timescale 1ns / 1ps
(* use_dsp48 = "no" *)
module Counter(input clk,input rst,output pulse);
     reg [23:0] cnt = 0;
     always @(posedge clk or posedge rst)
     begin
        if (rst)
        cnt <= 23'd0;
        else
            if(cnt >= 23'd499999)
            cnt <= 23'd0;
        else
            cnt <= cnt + 23'd1;
     end
    assign pulse =  (cnt == 23'd499999) ? 1'b1 : 1'b0;
endmodule
```



Resource	Utilization	Available	Utilization %
LUT	46	63400	0.07
FF	28	126800	0.02
IO	12	210	5.71
BUFG	2	32	6.25
MMCM	1	6	16.67

截图 2-lab9_2_1 资源使用

Number of BUFG/BUFGCTRL:______2______

Number of Slice LUTs used: ______46______

Number of FF used:______28______

Number of DSP48E1 slices used: ______0______

Number of IOs used:______12______

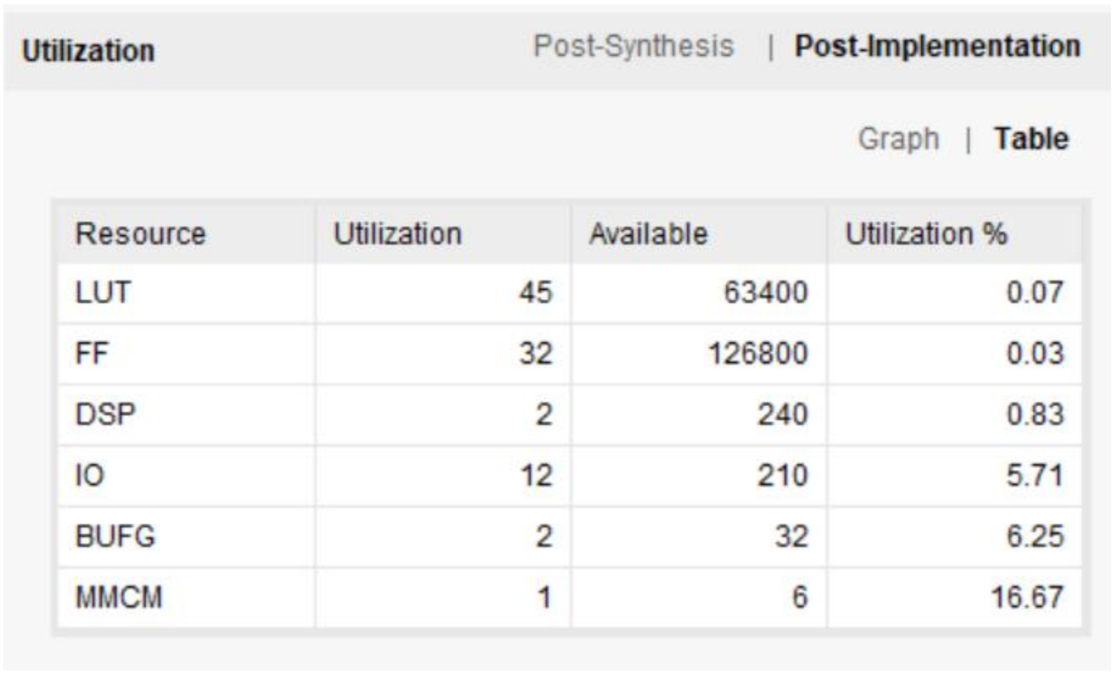
Lab9_2_2

代码 up_down_Counter.v

```
`timescale 1ns / 1ps
 (* use_dsp48 = "yes" *)
module up_down_Counter #(parameter COUNT_SIZE = 8) (
    input CLK100MHZ,
    input up_down,
    input enable,
    input reset,
    output reg [COUNT_SIZE - 1:0] Count
    );

    wire pulse,clk;
    clk_wiz_0 CLOCK5M (clk,CLK100MHZ);
     Counter CLOCK (clk,~enable,pulse);
    always@(posedge pulse)
    begin
        if (up_down)
        begin
            if ((Count >= (1 << COUNT_SIZE) - 1) || (reset))
                Count   <= 'b0;
            else
                Count   <= Count + 'b1;
        end
        else
        begin
            if ((Count == 'b0) || (reset))
                Count   <= (1 << COUNT_SIZE) - 1;
            else
                Count   <= Count - 'b1;
        end
    end

endmodule
```

代码 Counter.v

```
`timescale 1ns / 1ps
(* use_dsp48 = "yes" *)
module Counter(input clk,input rst,output pulse);
     reg [23:0] cnt = 0;
     always @(posedge clk or posedge rst)
     begin
        if (rst)
        cnt <= 23'd0;
        else
            if(cnt >= 23'd499999)
            cnt <= 23'd0;
        else
            cnt <= cnt + 23'd1;
     end
    assign pulse =  (cnt == 23'd499999) ? 1'b1 : 1'b0;
endmodule
```


Utilization Post-Synthesis | Post-Implementation

Graph | Table

Resource	Utilization	Available	Utilization %
LUT	45	63400	0.07
FF	32	126800	0.03
DSP	2	240	0.83
IO	12	210	5.71
BUFG	2	32	6.25
MMCM	1	6	16.67

截图 3 lab9_2_2 资源使用

Number of BUFG/BUFGCTRL:____2____

Number of Slice LUTs used: ____45____

Number of FF used:____32____

Number of DSP48E1 slices used: ____2____

Number of IOs used:____12____

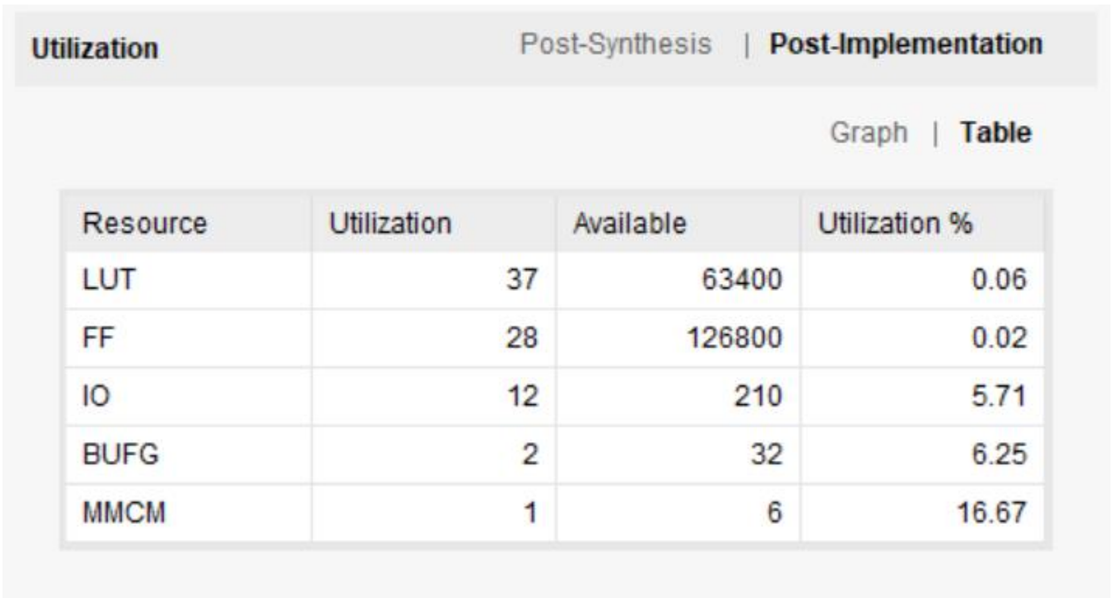
Lab9_2_3

代码 up_down_Counter.v

```
`timescale 1ns / 1ps
module up_down_Counter #(parameter COUNT_SIZE = 8) (
    input CLK100MHZ,
    input up_down,
    input enable,
    input reset,
    output [COUNT_SIZE - 1:0] Count
    );
    wire pulse,clk;
    clk_wiz_0 CLOCK5M (clk,CLK100MHZ);
    Counter CLOCK (clk,~enable,pulse);
    c_counter_binary_0 UpCounter (pulse,enable,reset,up_down,Count);
endmodule
```

代码 Counter.v

```
`timescale 1ns / 1ps
module Counter(input clk,input rst,output pulse);
     reg [23:0] cnt = 0;
     always @(posedge clk or posedge rst)
     begin
        if (rst)
        cnt <= 23'd0;
        else
            if(cnt >= 23'd499999)
            cnt <= 23'd0;
        else
            cnt <= cnt + 23'd1;
     end
    assign pulse =  (cnt == 23'd499999) ? 1'b1 : 1'b0;
endmodule
```



Utilization   Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	37	63400	0.06
FF	28	126800	0.02
IO	12	210	5.71
BUFG	2	32	6.25
MMCM	1	6	16.67

截图 4-lab9_2_3 资源使用

Number of BUFG/BUFGCTRL:______2______

Number of Slice LUTs used: ______37______

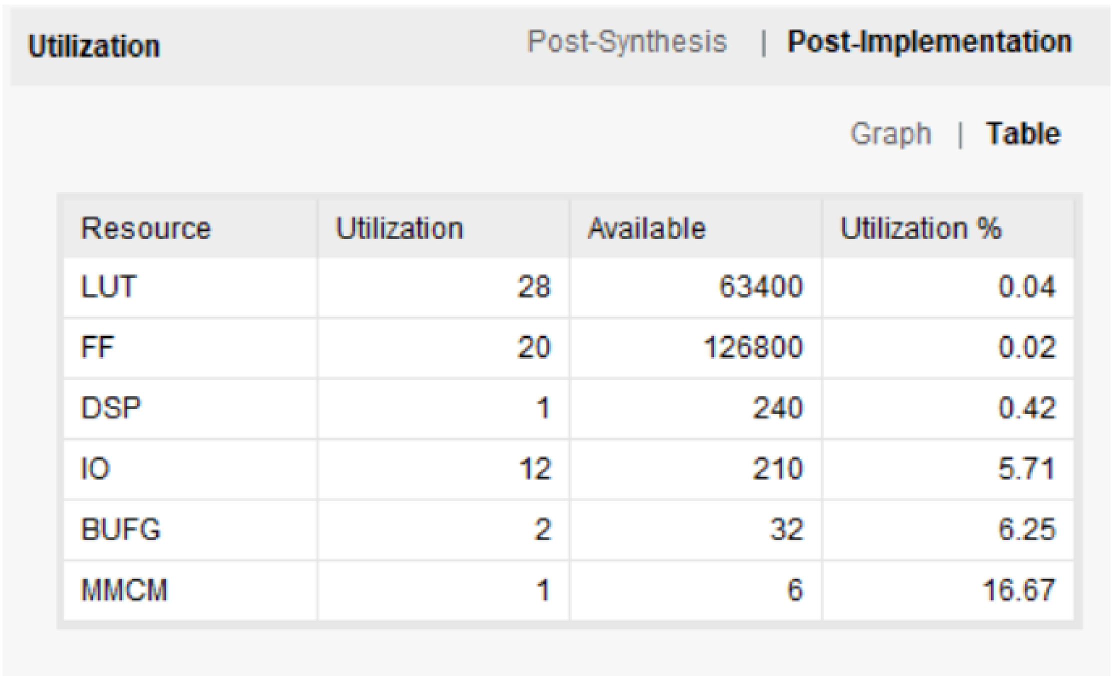
Number of FF used:______28______

Number of DSP48E1 slices used: ______0______

Number of IOs used:______12______

Lab9_2_4

代码与 Lab9_2_3 相同



Utilization — Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	28	63400	0.04
FF	20	126800	0.02
DSP	1	240	0.42
IO	12	210	5.71
BUFG	2	32	6.25
MMCM	1	6	16.67

截图 5-lab9_2_4 资源使用

Number of BUFG/BUFGCTRL:______2______

Number of Slice LUTs used: _____28_____

Number of FF used:_____20______

Number of DSP48E1 slices used: _____1_____

Number of IOs used:_____12_____