实验题目：计数器设计

班级：无 42

学号：2014011058

姓名：陈誉博

日期：2016 年 4 月 15 日

一、 实验目的

1．掌握简单时序逻辑电路的设计方法；
2．理解同步计数器和异步计数器的原理；
3．了解任意进制计数器的设计方法。

二、 实验原理及设计思路

计数器的原理：

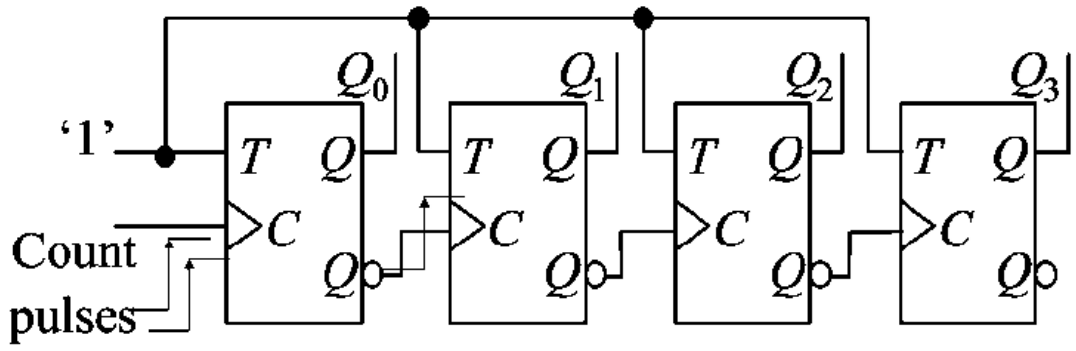
计数器是一种常用的时序电路，按照规定的方式改变内部各个触发器的状态，以记录输入的时钟脉冲的个数。按照规定的计数顺序的不同，计数器可以分为加法计数器、减法计数器、可逆计数器和不同进制的计数器；按照工作方式的不同，可以分为异步计数器和同步计数器。加法计数器在计数脉冲依次输入时，相应的二进制数据是依次增长的。每来一个计数脉冲，最低位的状态变化一次，其后各位则在低一位触发器的状态从 1 变为 0 时发生变化。这样，利用低一位的反相输入作为高一位的始终输入，便可以构成异步加法计数器。减法计数器的计数规律与加法计数器相反，每来一个计数脉冲，计数器的值减一。利用 D 触发器可以方便的构成减法计数器。与加法计数器不同的是，减法计数器利用低一位的输出作为高一位的时钟输入，而加法计数器则是利用低一位的反向输出作为高一位的时钟输入。

这两种计数器都属于异步计数器，由于计数控制信号是在各级之间逐级传递，这种计数器从时钟脉冲上升沿打到最后一个触发器翻转到规定的状态需要较长时间的延时，计数器位数越多，翻转到稳定状态的时间越长。为了提高计数器的工作速度，可以采用同步计数器。在同步计数器中，各个触发器使用同一个计数控制时钟，每一位在时钟上升沿到来时是否反转取决于比其低的位是否都为 1.其中，触发器的反转是在时钟上升沿同步进行的，其翻转稳定时间仅仅取决于单级触发器的翻转时间，与计数器的位数无关。

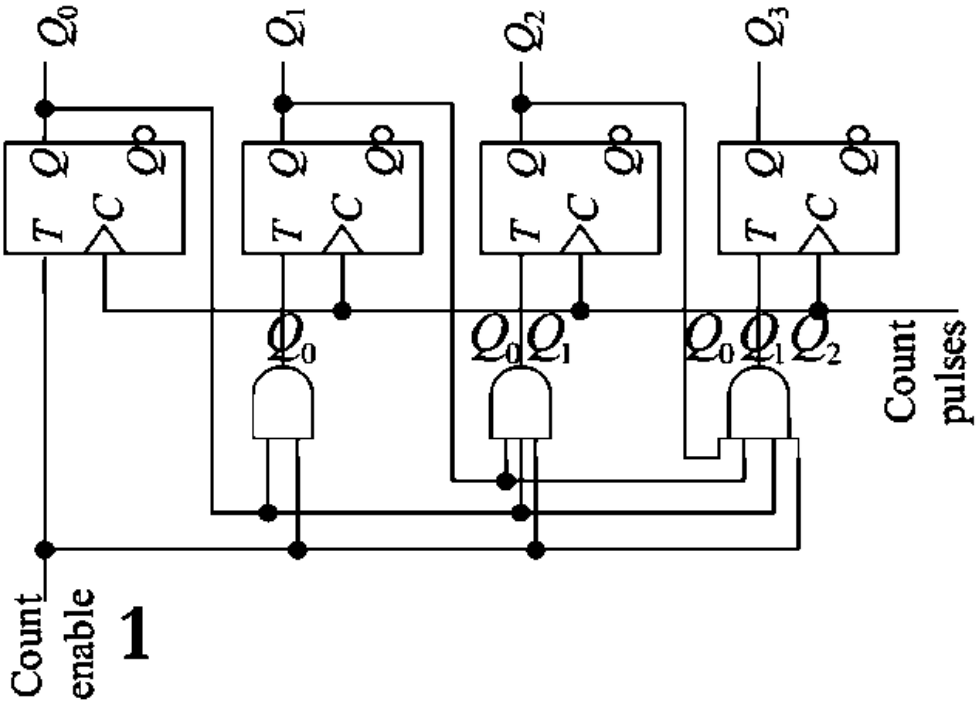
此次实验要求设计三种不同的计数器：

1) 具有异步复位控制的 4bits 二进制同步加法计数器。
2) 具有异步复位控制的 4bits 二进制异步加法计数器。
3) 在上述计数器的基础上设计一个减法计数器。

异步计数器的电路图如下：



同步计数器的电路图如下：



此次实验设计中减法计数器采用异步控制，只需要将异步加法计数器中的逻辑翻转，触发条件由次一级的下降沿改为上升沿即可，此处不再给出电路图。

三、 关键代码及源文件清单

1. 公共代码

1) 四位计数码转七段数码管控制代码 bcd7.v：

```
1  `timescale 1ns/1ns
2  module BCD7(din,dout);
3  input [3:0] din;
4  output [6:0] dout;
5  assign dout=(din==4'h0)?7'b111_1110:
6              (din==4'h1)?7'b011_0000:
7              (din==4'h2)?7'b110_1101:
8              (din==4'h3)?7'b111_1001:
9              (din==4'h4)?7'b011_0011:
10             (din==4'h5)?7'b101_1011:
11             (din==4'h6)?7'b101_1111:
12             (din==4'h7)?7'b111_0000:
13             (din==4'h8)?7'b111_1111:
14             (din==4'h9)?7'b111_1011:
15             (din==4'hA)?7'b111_0111:
16             (din==4'hB)?7'b001_1111:
17             (din==4'hC)?7'b100_1110:
18             (din==4'hD)?7'b011_1101:
19             (din==4'hE)?7'b100_1111:
20             (din==4'hF)?7'b100_0111:7'b0;
21 endmodule
```

2) 按键除抖代码 debounce.v:

```
module debounce(clk,key_i,key_o);
    input clk;
    input key_i;
    output key_o;

    parameter NUMBER = 24'd10_000_000;
    parameter NBITS = 24;

    reg [NBITS-1:0] count;
    reg key_o_temp;

    reg key_m;
        reg key_i_t1,key_i_t2;

    assign key_o = key_o_temp;

        always @ (posedge clk) begin
                key_i_t1 <= key_i;
                key_i_t2 <= key_i_t1;
        end

    always @ (posedge clk) begin
        if (key_m!=key_i_t2) begin
            key_m <= key_i_t2;
            count <= 0;
        end
        else if (count == NUMBER) begin
            key_o_temp <= key_m;
        end
        else count <= count+1;
    end
endmodule
```

3) 端口约束文件 bcd7.xdc:

```
set_property PACKAGE_PIN V17 [get_ports {reset}]
set_property PACKAGE_PIN U17 [get_ports {anjian}]

set_property PACKAGE_PIN W7 [get_ports {shuchu[6]}]
set_property PACKAGE_PIN W6 [get_ports {shuchu[5]}]
set_property PACKAGE_PIN U8 [get_ports {shuchu[4]}]
set_property PACKAGE_PIN V8 [get_ports {shuchu[3]}]
set_property PACKAGE_PIN U5 [get_ports {shuchu[2]}]
set_property PACKAGE_PIN V5 [get_ports {shuchu[1]}]
set_property PACKAGE_PIN U7 [get_ports {shuchu[0]}]

set_property PACKAGE_PIN W5 [get_ports clk]

create_clock -period 10.000 -name CLK -waveform {0.000 5.000} [get_ports clk]
#set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets clk]

set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports {reset}]
set_property IOSTANDARD LVCMOS33 [get_ports {anjian}]
set_property IOSTANDARD LVCMOS33 [get_ports {shuchu[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {shuchu[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {shuchu[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {shuchu[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {shuchu[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {shuchu[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {shuchu[0]}]
```

2. 计数器逻辑代码

1) 异步加法计数器 yibu.v

```
`timescale 1ns/1ns
module yibu_counter(clk,anjian,reset,shuchu);
input clk,anjian,reset;
output [6:0]shuchu;
reg [3:0]q_out;
wire [6:0]bcd7_out;
wire chudou;
debounce db(.clk(clk),.key_i(anjian),.key_o(chudou));
always @(posedge chudou or negedge reset)
begin
        if(~reset) q_out[0]<=1'b0;
        else q_out[0]<=~q_out[0];
end
always @(negedge reset or negedge q_out[0])
begin
        if(~reset) q_out[1]<=1'b0;
        else q_out[1]<=~q_out[1];
end
always @(negedge reset or negedge q_out[1])
begin
        if(~reset) q_out[2]<=1'b0;
        else q_out[2]<=~q_out[2];
end
always @(negedge reset or negedge q_out[2])
begin
        if(~reset) q_out[3]<=1'b0;
        else q_out[3]<=~q_out[3];
end
BCD7 b1(.din(q_out),.dout(bcd7_out));
assign shuchu=~bcd7_out;
endmodule
```

2) 同步加法计数器 tongbu.v:

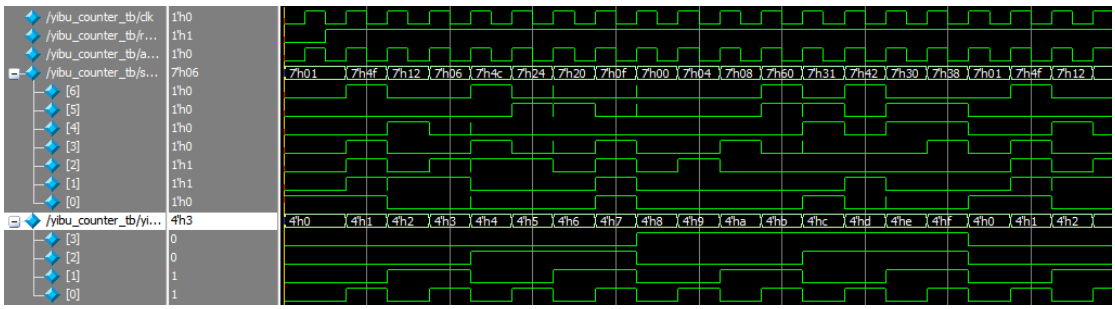
```
`timescale 1ns/1ns
module tongbu_counter(clk,anjian,reset,shuchu);
input clk,anjian,reset;
output [6:0]shuchu;
reg [3:0]q_out;
wire [6:0]bcd7_out;
wire chudou;
debounce db(.clk(clk),.key_i(anjian),.key_o(chudou));
always @(posedge chudou or negedge reset)
begin
        if(~reset) begin
                q_out[0]<=1'b0;
                q_out[1]<=1'b0;
                q_out[2]<=1'b0;
                q_out[3]<=1'b0;
        end
        else  begin
                q_out[0]<=~q_out[0];
                if(q_out[0]==1'b1) q_out[1]<=~q_out[1];
                if(q_out[0]==1'b1&&q_out[1]==1'b1) q_out[2]<=~q_out[2];
                if(q_out[0]==1'b1&&q_out[1]==1'b1&&q_out[2]==1'b1) q_out[3]<=~q_out[3];
                if(q_out[0]==1'b1&&q_out[1]==1'b1&&q_out[2]==1'b1&&q_out[3]==1'b1) q_out=4'b0;
        end
end
BCD7 b1(.din(q_out),.dout(bcd7_out));
assign shuchu=~bcd7_out;
endmodule
```

3) 异步减法计数器：jianfa.v

```
1  `timescale 1ns/1ns
2  module jianfa_counter(clk,anjian,reset,shuchu);
3  input clk,anjian,reset;
4  output [6:0]shuchu;
5  reg [3:0]q_out;
6  wire [6:0]bcd7_out;
7  wire chudou;
8  debounce db(.clk(clk),.key_i(anjian),.key_o(chudou));
9  always @(posedge chudou or negedge reset)
10 begin
11         if(~reset) q_out[0]<=1'b1;
12         else q_out[0]<=~q_out[0];
13 end
14 always @(negedge reset or posedge q_out[0])
15 begin
16         if(~reset) q_out[1]<=1'b1;
17         else q_out[1]<=~q_out[1];
18 end
19 always @(negedge reset or posedge q_out[1])
20 begin
21         if(~reset) q_out[2]<=1'b1;
22         else q_out[2]<=~q_out[2];
23 end
24 always @(negedge reset or posedge q_out[2])
25 begin
26         if(~reset) q_out[3]<=1'b1;
27         else q_out[3]<=~q_out[3];
28 end
29 BCD7 b1(.din(q_out),.dout(bcd7_out));
30 assign shuchu=~bcd7_out;
31 endmodule
```

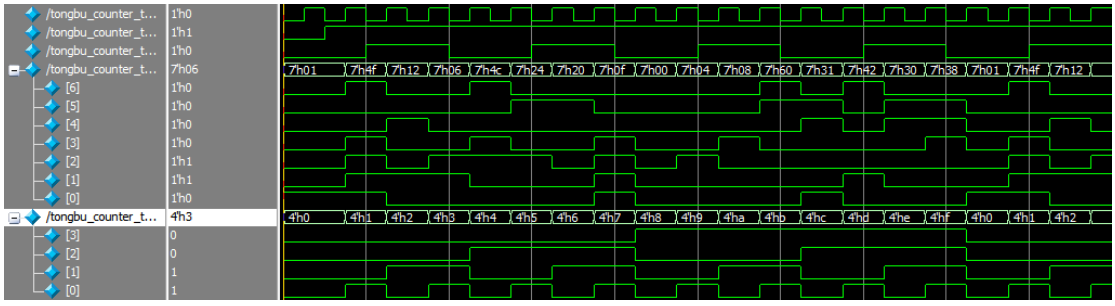
四、 仿真结果与分析

1. 异步加法计数器：



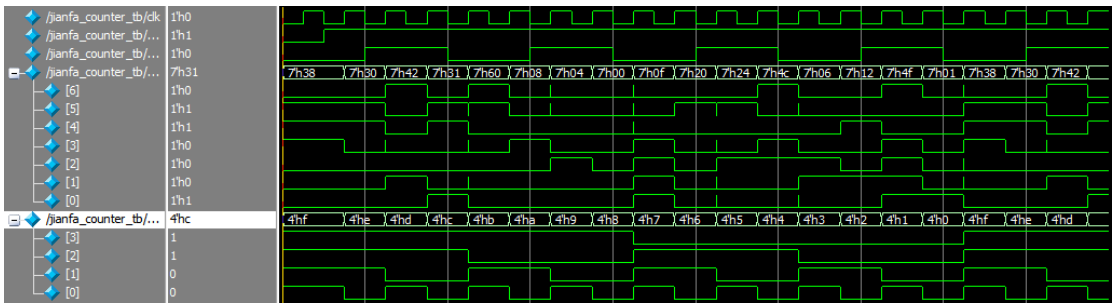
中间的[6:0]向量为七段数码管输出，最下的[4:0]向量为四位计数输出。

2. 同步加法计数器：



中间的[6:0]向量为七段数码管输出，最下的[4:0]向量为四位计数输出。

3. 异步减法计数器



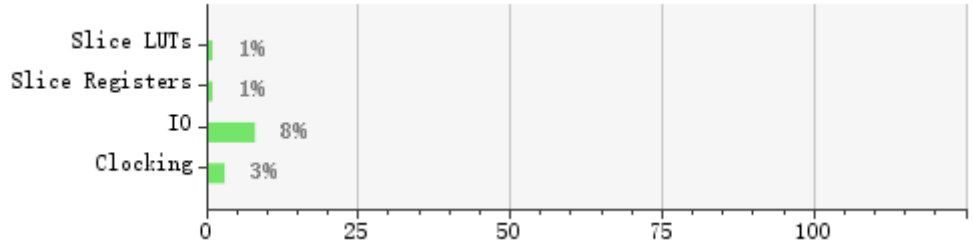
中间的[6:0]向量为七段数码管输出，最下的[4:0]向量为四位计数输出。

五、 综合报告

1. 异步加法计数器：

面积报告：

Resource	Utilization	Available	Utilizat...
Slice LUTs	16	20800	0.08
Slice Registers	32	41600	0.08
IO	10	120	8.33
Clocking	1	32	3.12



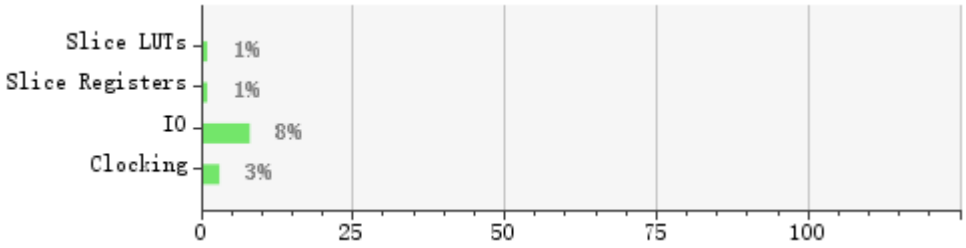
时序报告：

Name	Waveform	Period (ns)	Frequency (MHz)
CLK	{0.000 5.000}	10.000	100.000

2. 同步加法计数器：

面积报告：

Resource	Utilization	Available	Utilizat...
Slice LUTs	39	20800	0.19
Slice Registers	32	41600	0.08
IO	10	120	8.33
Clocking	1	32	3.12



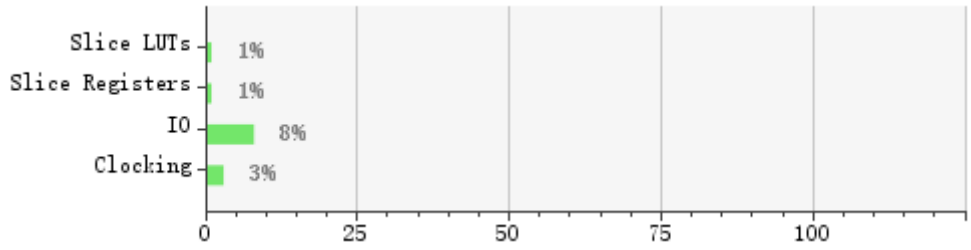
时序报告：

Name	Waveform	Period (ns)	Frequency (MHz)
CLK	{0.000 5.000}	10.000	100.000

3. 异步减法计数器：

面积报告：

Resource	Utilization	Available	Utilizat...
Slice LUTs	40	20800	0.19
Slice Registers	32	41600	0.08
IO	10	120	8.33
Clocking	1	32	3.12



时序报告：

Name	Waveform	Period (ns)	Frequency (MHz)
CLK	{0.000 5.000}	10.000	100.000

六、 硬件调试情况

1. 异步加法计数器

在实验板上，按键 BTND 为计数脉冲输入，拨动开关 SW0 为异步复位控制信号。在 SW0 拨动到高电平时，每按一次按键，七段数码管的数字加 1，从 0000-ffff，到 ffff 之后跳回 0000；在任何时刻，在 SW0 拨动到低电平的时刻数码管的显示变为 0000。

2. 同步加法计数器

同步加法计数器的情况和异步加法计数器基本相同。

3. 异步减法计数器

在 SW0 拨动到高电平时，每按一次按键，七段数码管的数字加 1，从 ffff-0000，到 0000 之后跳回 ffff；在任何时刻，在 SW0 拨动到低电平的时刻数码管的显示变为 ffff。

（注：由于除抖模块的作用，当 BTND 按键两次按下的时间间隔过小时，数码管的数字不发生变化）

七、 实验总结与体会

由于是首次实验，首次接触到 Verilog 行为级的编程，对于 Verilog 编程的各种功能、各种模块的作用以及编程中需要注意的一些问题都有了初步的理解。同时通过此次实验也掌握了分析程序的时序情况和面积情况的方法。此次实验对于熟悉、理解、学习 Verilog 编程有很大帮助。