

计算机体系结构实验

实验2报告书

作者：李珉超

学号：515030910361

2017.4.2

1 实验介绍

1.1 实验名称

FPGA基础实验2：4 bits binary counter with time divider

1.2 实验目的

- 1.掌握Xilinx逻辑设计工具ISE的设计流程
- 2.初步掌握使用VerilogHDL硬件描述语言进行简单的逻辑设计
- 3.掌握UCF(用户约束文件)的用法和作用
- 4.熟悉Xilinx Spartan 3E实验开发板
- 5.学习分频器的编写，了解同步信号的工作过程

1.3 实验范围

- 1.ISE 13.4 的使用
- 2.使用VerilogHDL进行逻辑设计
- 3.编辑UCF
- 4.iMPACT的使用
- 5.Spartan 3E实验板的使用

2 实验过程

2.1 counter

2.1.1 实验模块

mainClock是带宽为1的输入信号，reset是带宽为1的输入信号，count是带宽为4的输出信号。

在时钟上升沿，如果reset信号不是0，那么count自减1。如果reset信号是0，那么count的值初始化为4 ‘b1111。

always 模块不断检测条件。

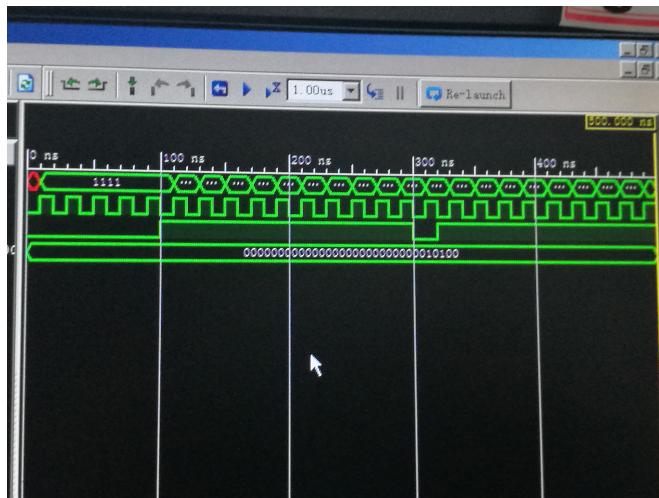
```
module counter(
    mainClock,
    reset,
    count
);
    input mainClock;
    input reset;
    output [3:0] count;
    reg [3:0] count;
    always @ (posedge slowClock)
        begin
            if(!reset)
                count <= 4'b1111;
            else
                count <= count - 1;
        end
endmodule
```

2.1.2 仿真

mainClock 和reset 初始化为0.
延迟100ns后， reset设置为1.
之后每200ns， reset值为1.
然后的20ns， reset值变为0。如此循环。

```
module countert;
    // Inputs
    reg mainClock;
    reg reset;
    // Outputs
    wire [3:0] count;
    parameter DELY=20;
    // Instantiate the Unit Under Test (UUT)
    counter uut (
        .mainClock(mainClock),
        .reset(reset),
        .count(count)
    );
    always #(DELY/2) mainClock = ~mainClock;
    initial begin
        // Initialize Inputs
        mainClock = 0;
        reset = 0;
        // Wait 100 ns for global reset to finish
        #100;
        // Add stimulus here
        reset=1'b1;
        #(DELY*10) reset=1'b0;
        #(DELY) reset=1'b1;
    end
endmodule
```

2.1.3 仿真图



2.2 divider

2.2.1 实验模块

```
module timerDivider(
    clockIn,
    clockOut
);
    input clockIn; output clockOut;
    reg clockOut;
    reg [23:0] buffer;

    always @ (posedge clockIn)
    begin
        buffer <= buffer + 1;
        clockOut <= & buffer;
    end
endmodule
```

```

module counter(
    mainClock,
    reset,
    count
);
    input mainClock;
    input reset;
    output [3:0] count;
    reg [3:0] count;
    wire slowClock;

    timerDivider td(.clockIn(mainClock), .clockOut(slowClock));

    always @ (posedge slowClock)
    begin
        if(!reset)
            count <= 4'b1111;
        else
            count <= count - 1;
    end
endmodule

```

2.2.2 管脚定义

```

NET "mainClock" LOC = "C9" | IOSTANDARD = LVCMOS33;
NET "count < 3 >" LOC = "F11" | IOSTANDARD = LVTTL | SLEW = SLOW |
DRIVE = 8;
NET "count < 2 >" LOC = "E11" | IOSTANDARD = LVTTL | SLEW = SLOW |
DRIVE = 8;

```

```

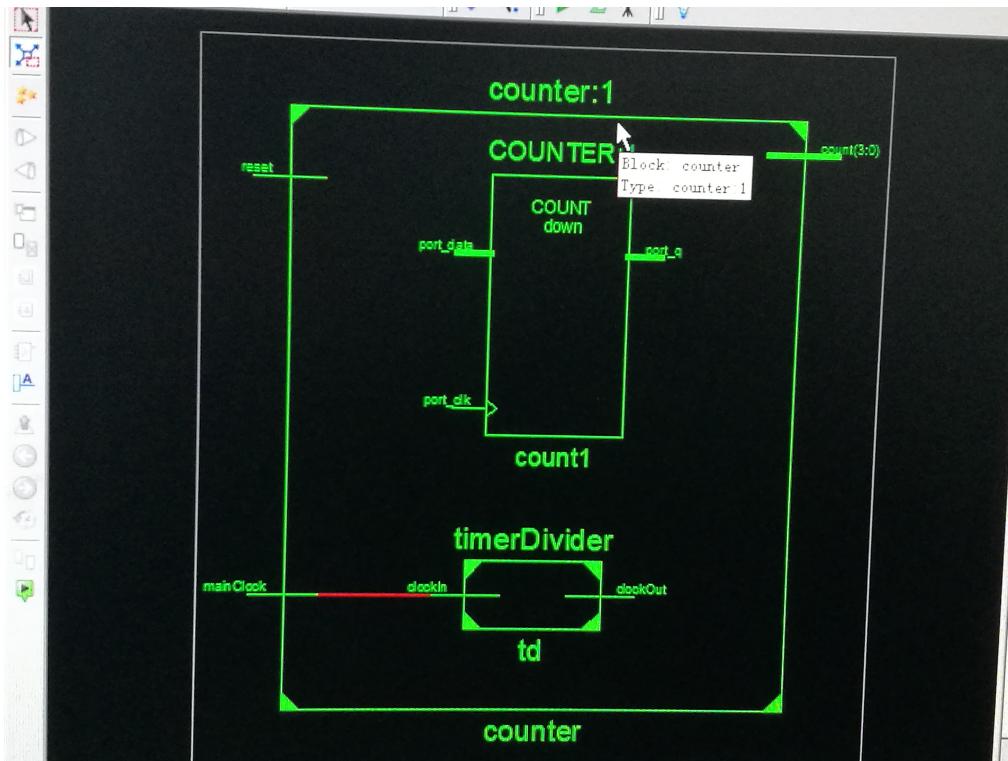
NET "count < 1 >" LOC = "E12" | IOSTANDARD = LVTTL | SLEW = SLOW |
DRIVE = 8;

NET "count < 0 >" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW |
DRIVE = 8;

NET "reset" LOC = "L13" | IOSTANDARD = LVTTL | PULLUP;

```

2.2.3 电路逻辑



3 实验心得

了解了always模块的使用方法和posedge, negedge的使用方法。

学会了在仿真中，使用always # DELAY clk = ~clk 进行时钟信号的方法。

学会了在顶层模块中，调用子模块的方法，以及wire变量的使用。