

本科实验报告

课程名称： **FPGA应用**

主讲教师： 夏书峰

学院、系： 微电子学院

专 业： 电子科学与技术类

班 级：

学 号：

姓 名：

联系电话：

电子邮箱：

2021～2022学年第3学期

实验成绩记录表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 内容 | 学时 | 满分 | 得分 |
| 1 | Vivado/ModelSim实践 | 4 | 15 |  |
| 2 | 时序逻辑设计1 — RTC | 4 | 20 |  |
| 3 | Xilinx嵌入式逻辑分析仪 | 4 | 15 |  |
| 4 | Xilinx嵌入式处理器实践 | 4 | 20 |  |
| 5 | 时序逻辑设计2 — FSM | 8 | 30 |  |
|  |  |  |  |  |
| 总计 | | 24 | 100 |  |

主讲教师签字：

实验三《Xilinx ChipScope Pro嵌入式逻辑分析仪实践》实验报告

【设计文件与实验过程】

将Verilog HDL文件添加到Vivado工程，测试向量也一并添加到工程目录， 使用课件中用到的状态机逻辑作为基础设计文件。

【源文件divider.v】

`timescale 1ns / 1ps

module divider(sysclk, rst, lfclk );

input sysclk, rst;

output lfclk;

reg [7: 0]counter;

reg lfclk;

parameter divnum = 25 - 1;

always @ (posedge sysclk) begin

    if (rst) begin

        counter <= 0;

        lfclk <= 0;

    end

    else begin

        if (counter >= divnum) begin

            counter <= 0;

            lfclk <= ~lfclk;

        end

        else

            counter <= counter + 1;

    end

end

endmodule

修改【fsm.v】和【fsm\_vio\_top.v】如下，注释vio\_0模块例化，只在顶层模块中例化vio模块

【源文件fsm.v】

`timescale 1ns / 1ps

module fsm(sysclk, rst,  key, trigger, data);

input sysclk, rst , key;

output trigger;

output [5: 0] data;

wire key; //for vio test

reg [5: 0] data;

reg [2: 0] state; //state register

reg trigger; //signal to trigger logic analyzer

reg lfclk\_dly; //delay of lfclk (1 sysclk)

reg re, fe; //indicate rising edge or falling edge of lfclk

wire lfclk; //low freq. signal from divider block

divider div\_inst(.sysclk(sysclk), .rst(rst), .lfclk(lfclk));

// vio\_0 vio\_inst(.clk(sysclk), .probe\_in0(data[0]), .probe\_in1(data[1]),

//                .probe\_in2(data[2]), .probe\_in3(data[3]), .probe\_in4(data[4]),

//                .probe\_in5(data[5]), .probe\_out0(key) );

parameter st0 = 0, st1 = 1, st2 = 2, st3 = 3;

//check rising edge and falling edge of lfclk

always @(posedge sysclk) begin

    if (rst) begin

        lfclk\_dly <= 0;

        re <= 0;

        fe <= 0;

    end

    else begin

        lfclk\_dly <= lfclk;

        if ((lfclk\_dly == 0) && (lfclk == 1)) begin

            re <= 1;

            fe <= 0;

        end

        else if ((lfclk\_dly == 1) && (lfclk == 0)) begin

            re <= 0;

            fe <= 1;

        end

        else begin

            re <= 0;

            fe <= 0;

        end

    end

end

always @(posedge sysclk) begin

    if (rst) begin

        state <= 0;

        trigger <= 0;

        data <= 0;

    end

    else begin

        case (state)

            st0: begin

                if (key)

                    state <= st1;

                else

                    state <= st0;

            end

            st1: begin

                if (re) begin

                    trigger <= 1;

                    state <= st2;

                end

            end

            st2: begin

                trigger <= 0;

                if (fe) begin

                    data <= data + 1;

                    if (data > 60)

                        state <= st3;

                    else

                        state <= st1;

                end

            end

            st3: begin

                state <= st3;

            end

            default:

                state <= st0;

        endcase

    end

end

endmodule

【源文件fsm\_vio\_top.v】

`timescale 1ns / 1ps

module fsm\_vio\_top(sysclk, rst, trigger, data);

input sysclk;

input rst;

//input key;

output trigger;

output [5: 0]data;

wire key;

fsm u1(

        .sysclk(sysclk),

        .rst(rst),

        .key(key),

        .trigger(trigger),

        .data(data)

    );

vio\_0 u2(

          .clk(sysclk),

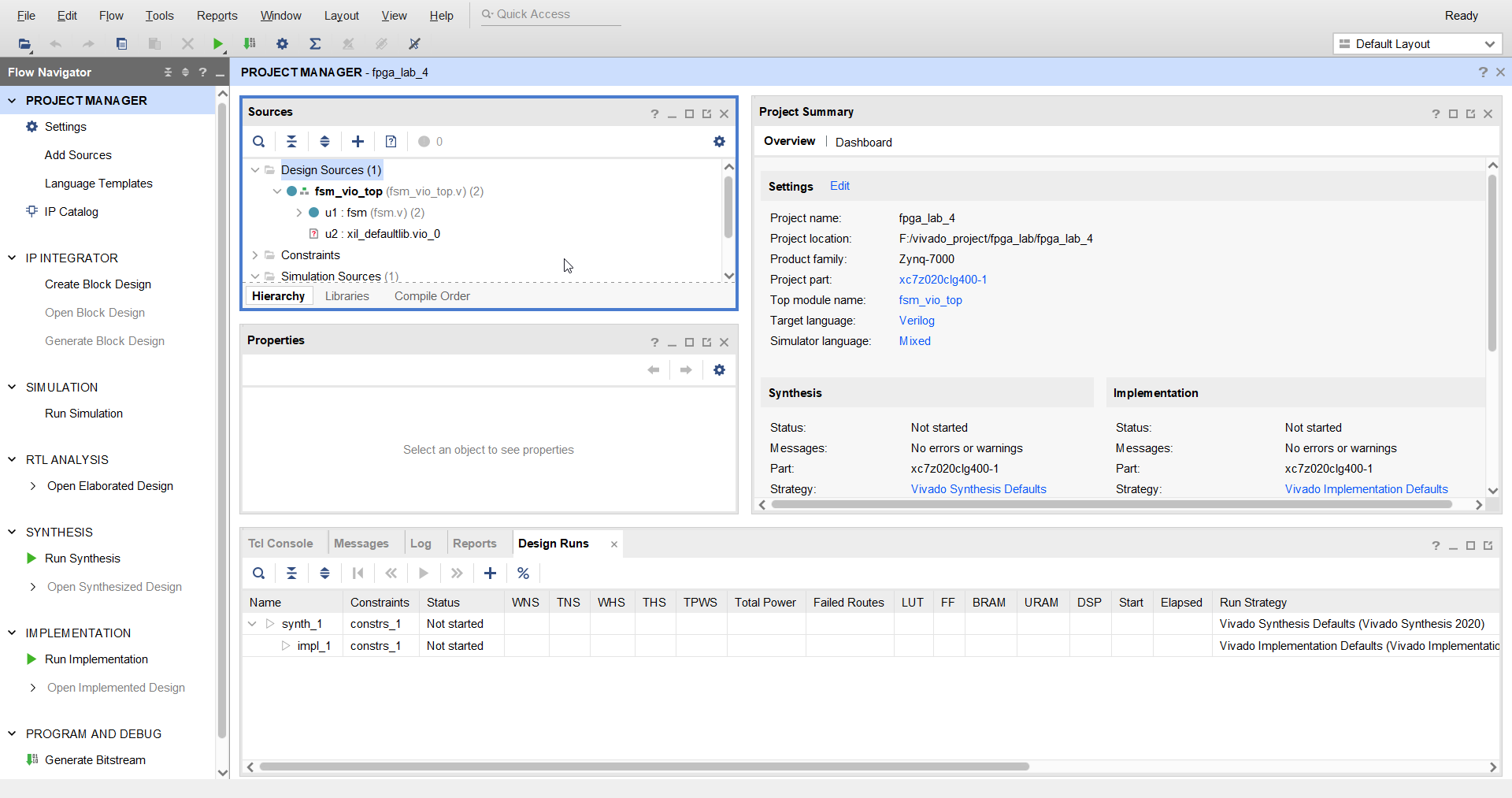
          .probe\_out0(key),

          .probe\_in0(data)

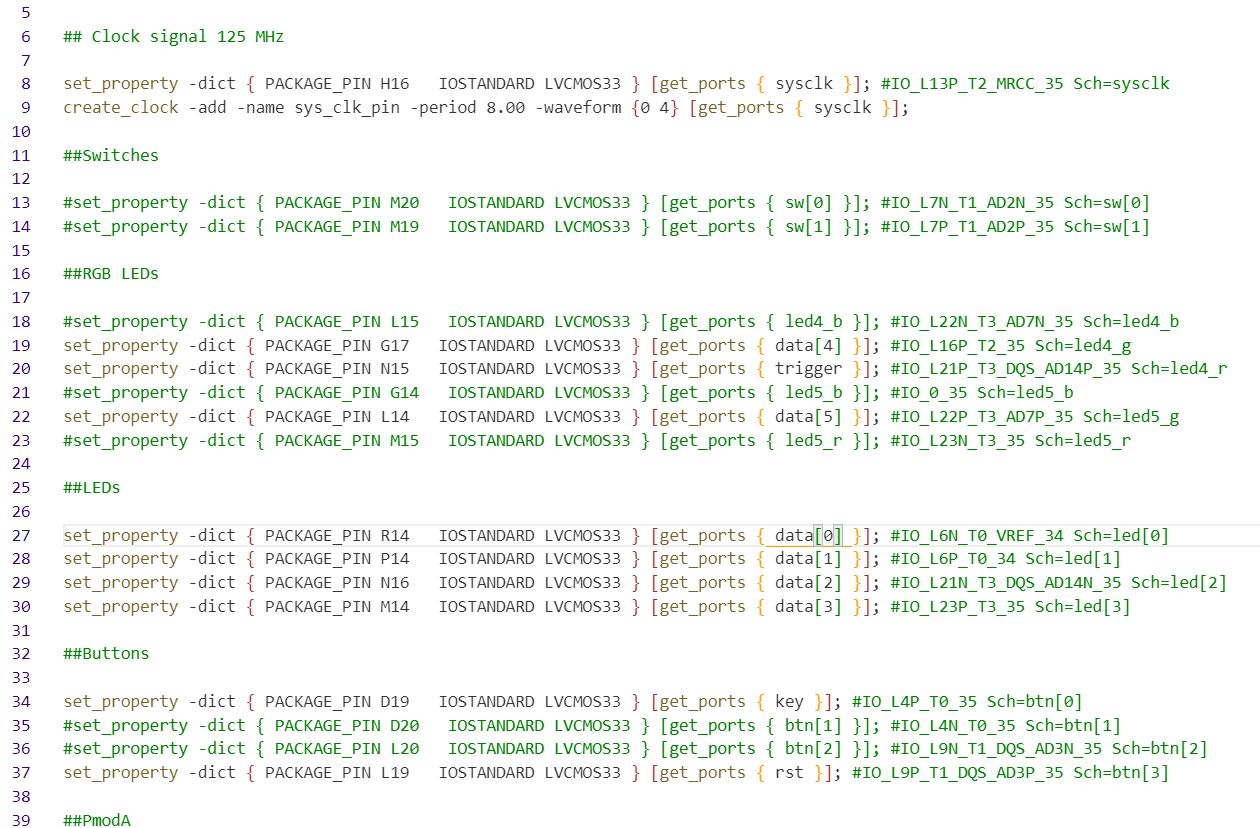
      );

endmodule

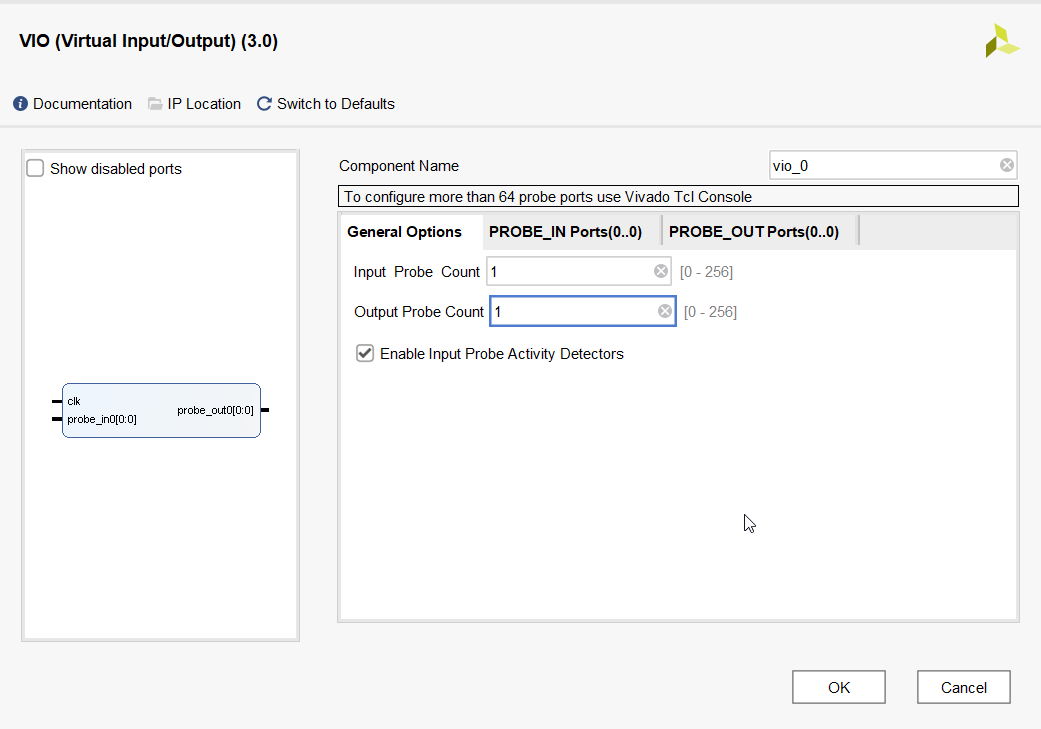
得到如下所示的目录

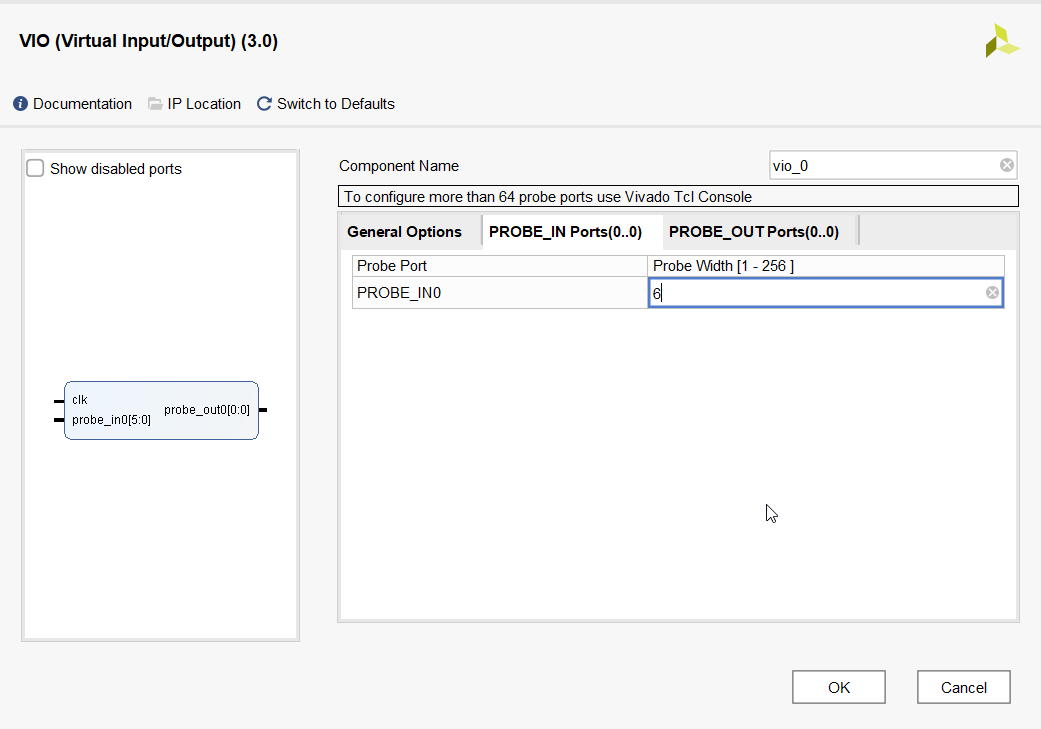


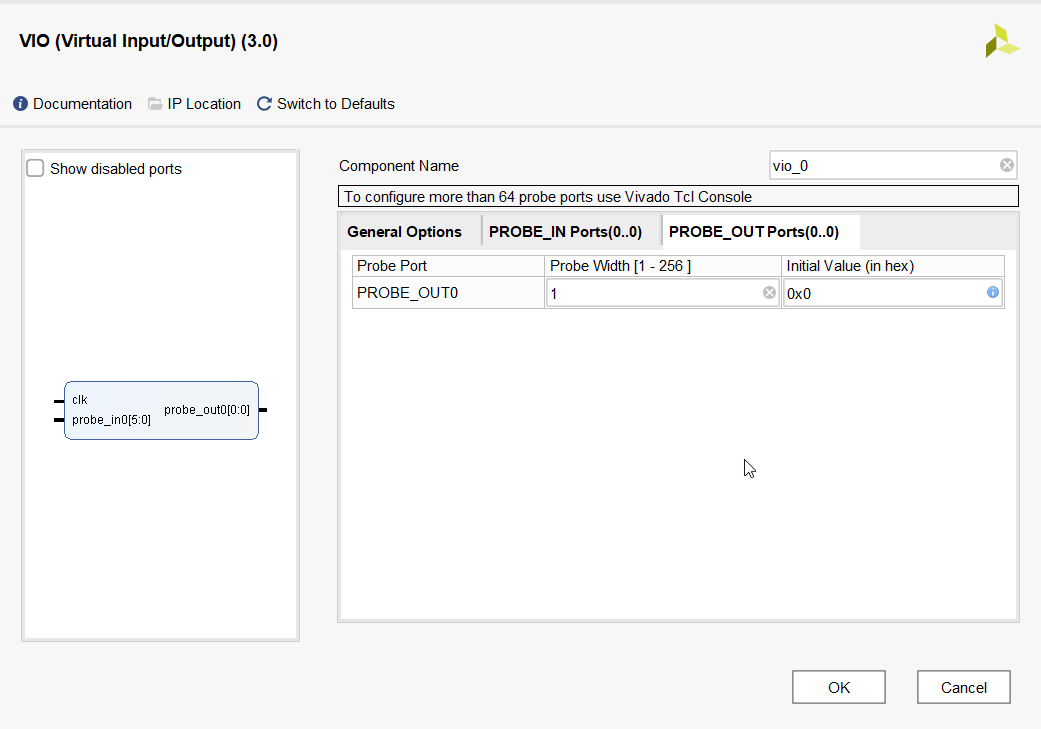
将官方XDC约束文件加入工程，修改如下：



打开Vivado中IP Catalog后生成vio实例，配置如下所示

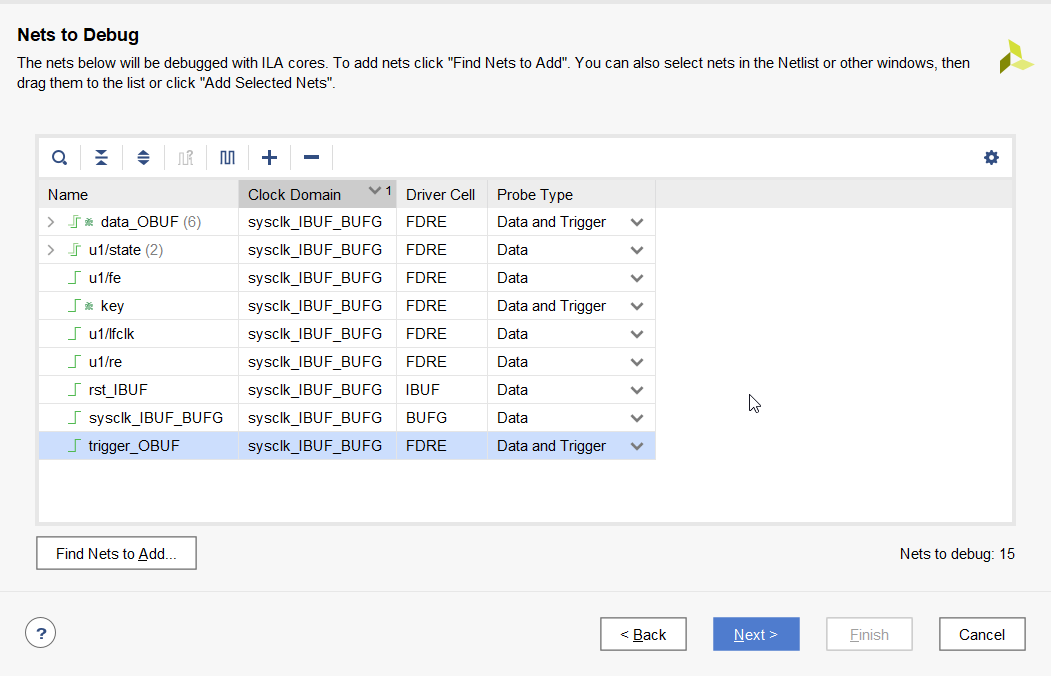


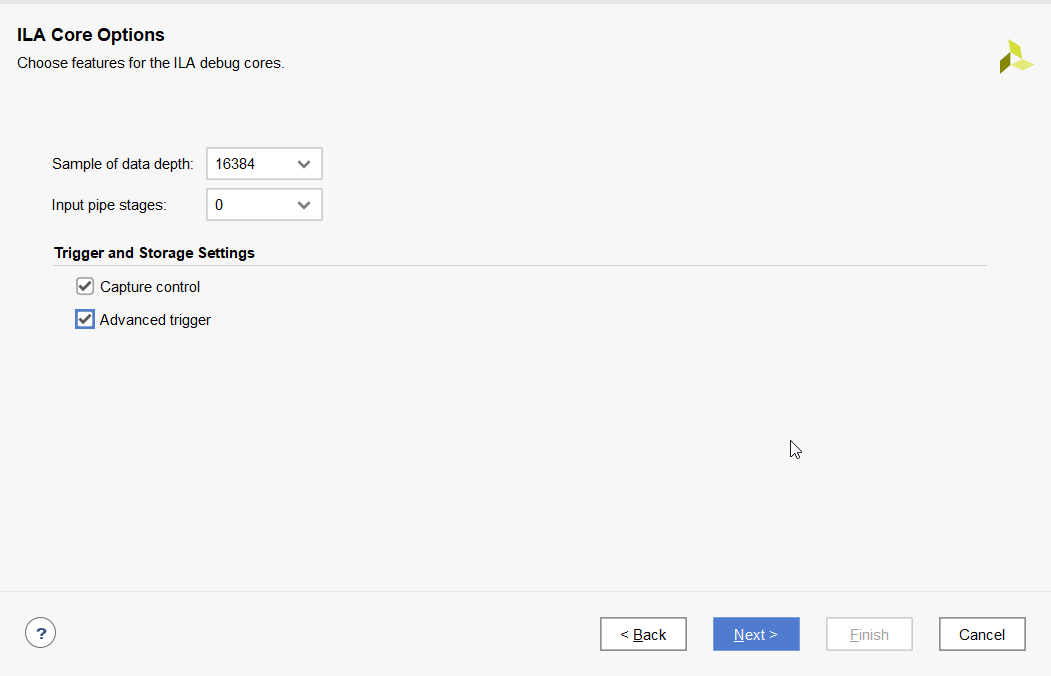




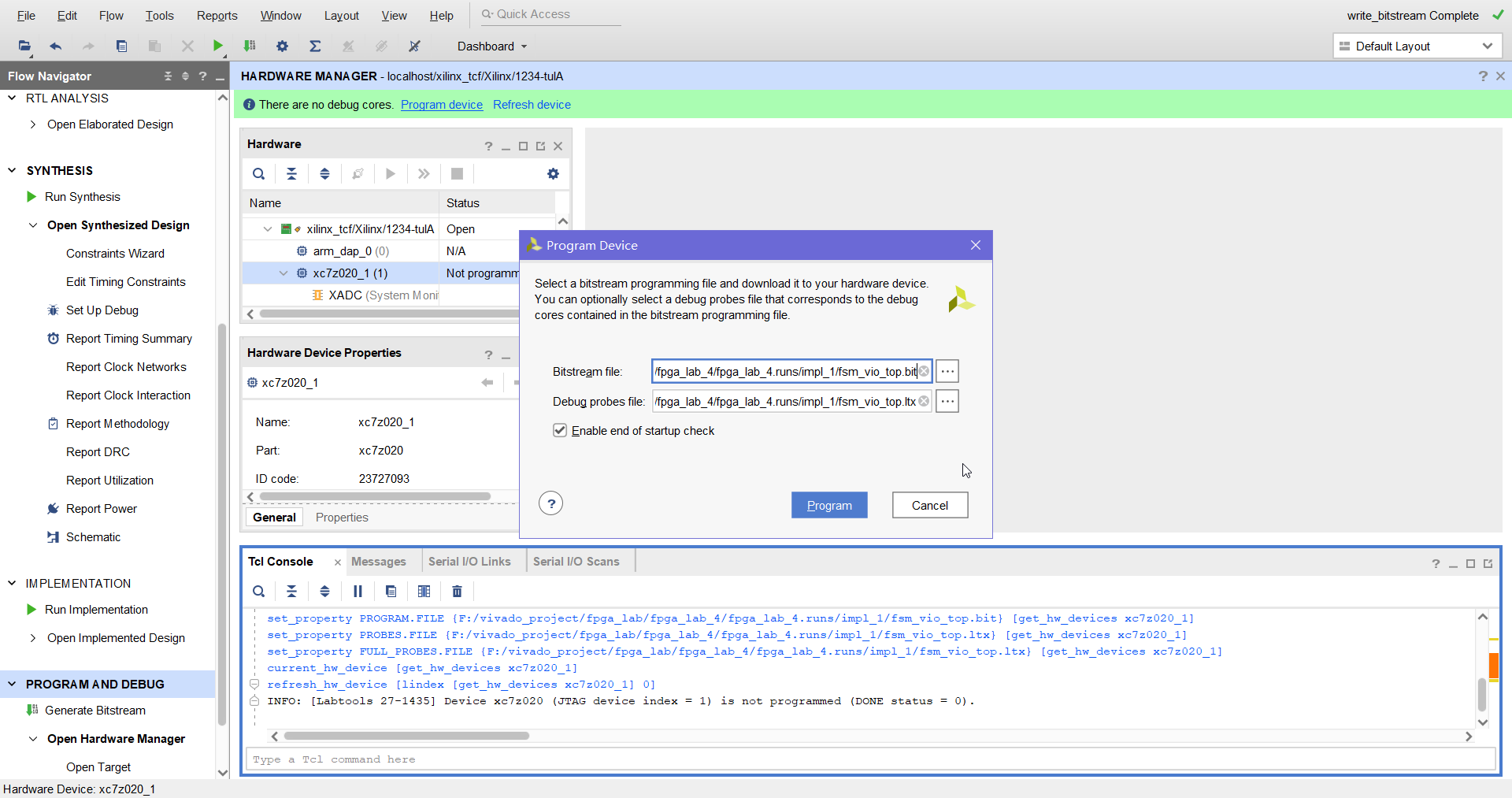
进行综合，完成后在左侧Set Debug部分设置调用ILA，选择查看网络data、state、fe、re、sysclk、lfclk、key、trigger，如下所示

设置时钟域为sysclk的BUFG生成的时钟，并设置触发信号。

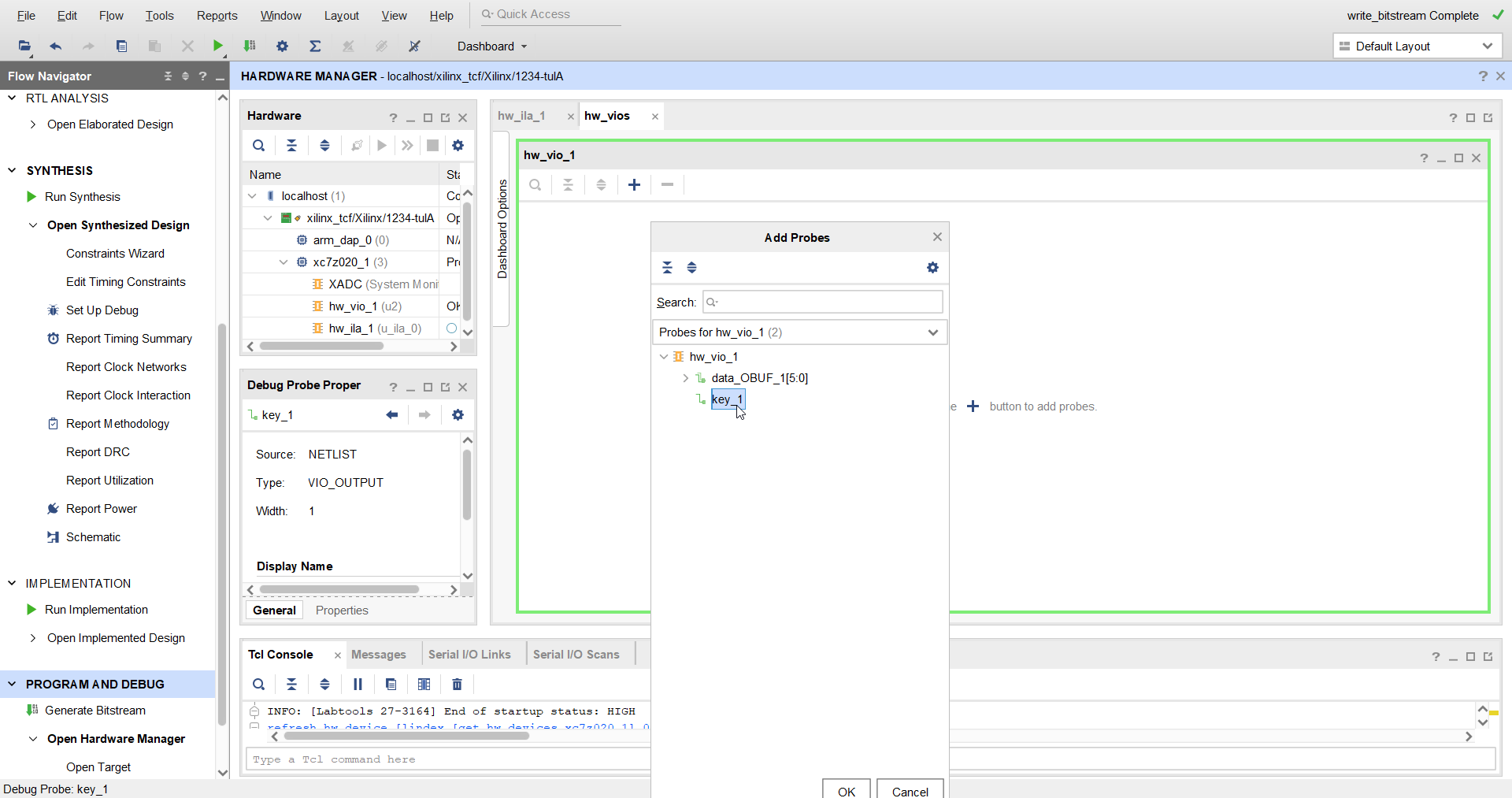




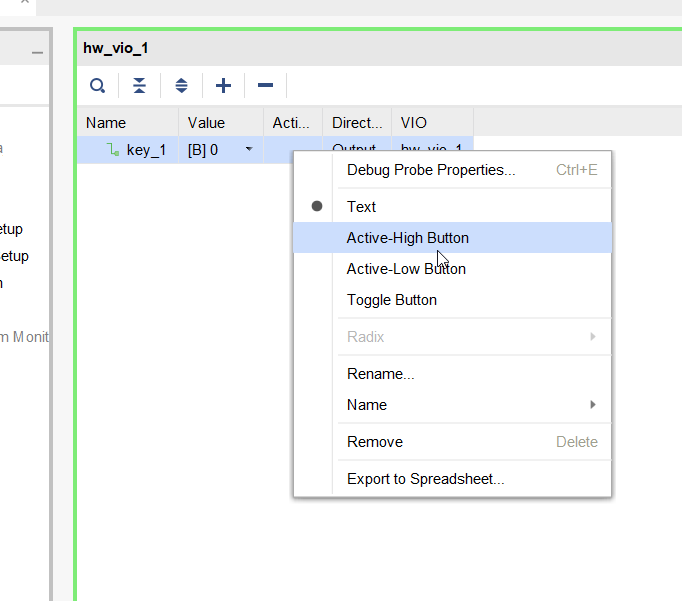
完成配置后进行布局布线和生成比特流，加载到开发板设备中。完成烧录后，右侧出现调试界面，在hw\_vio\_1面板中设置要调试的数据线

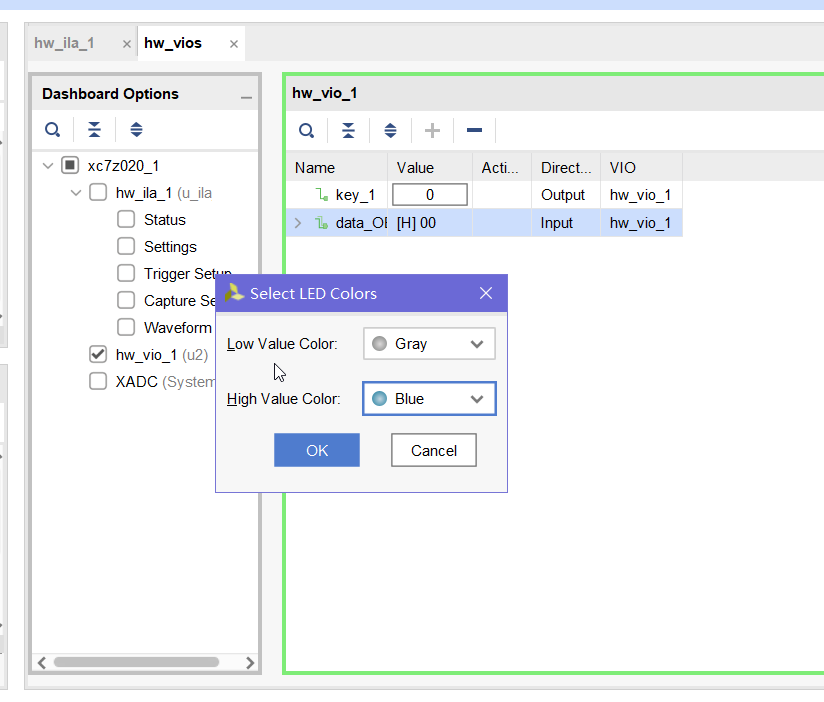


将需要输入输出的数据线加入vio中，如下

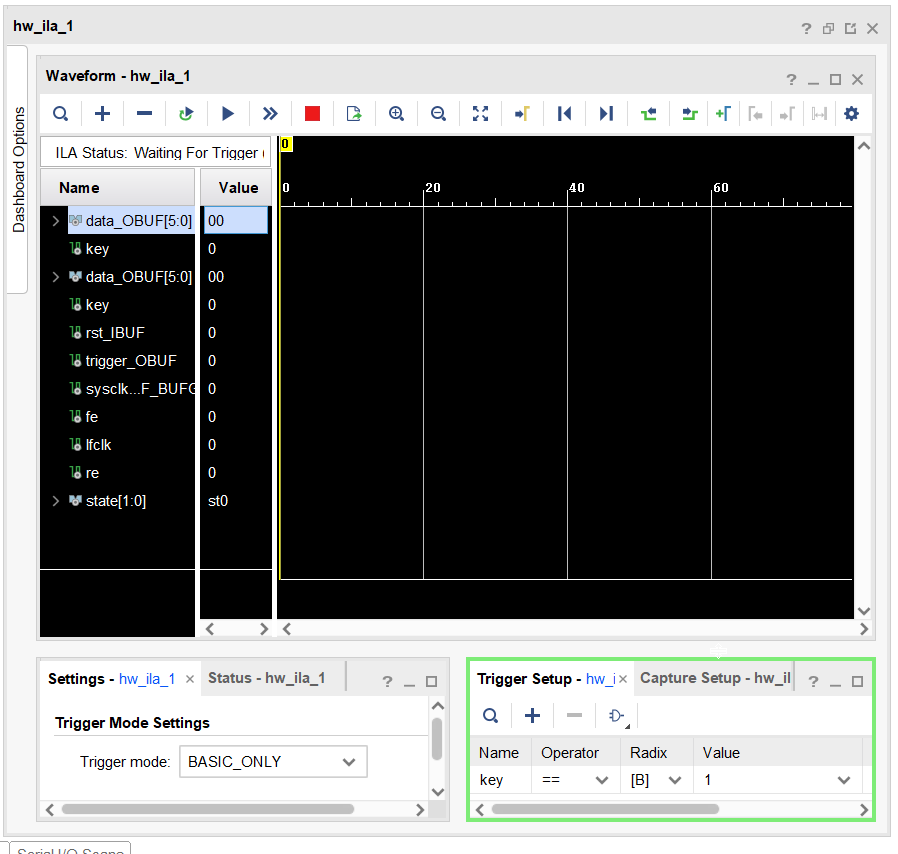


将key设置为按钮模式，将data显示设置为led模式

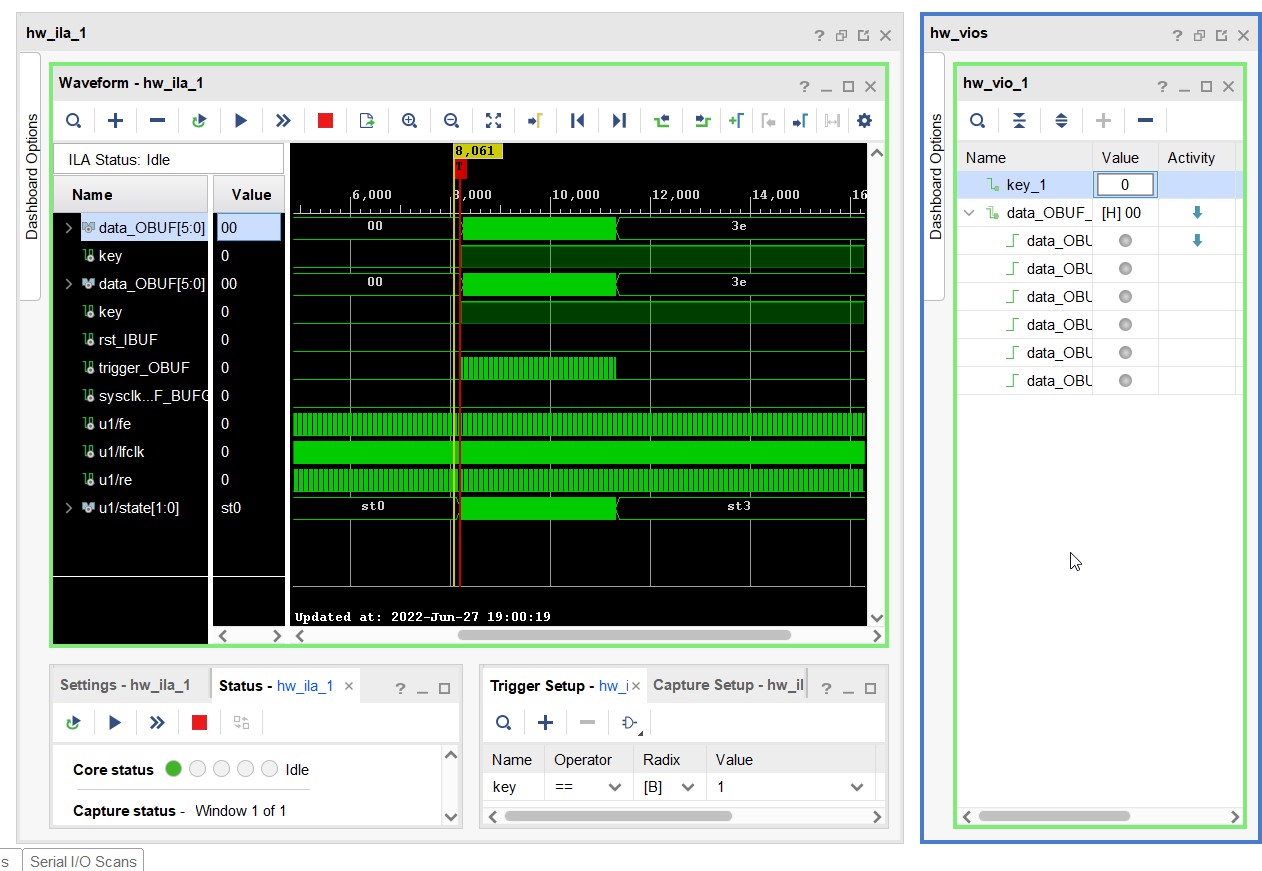




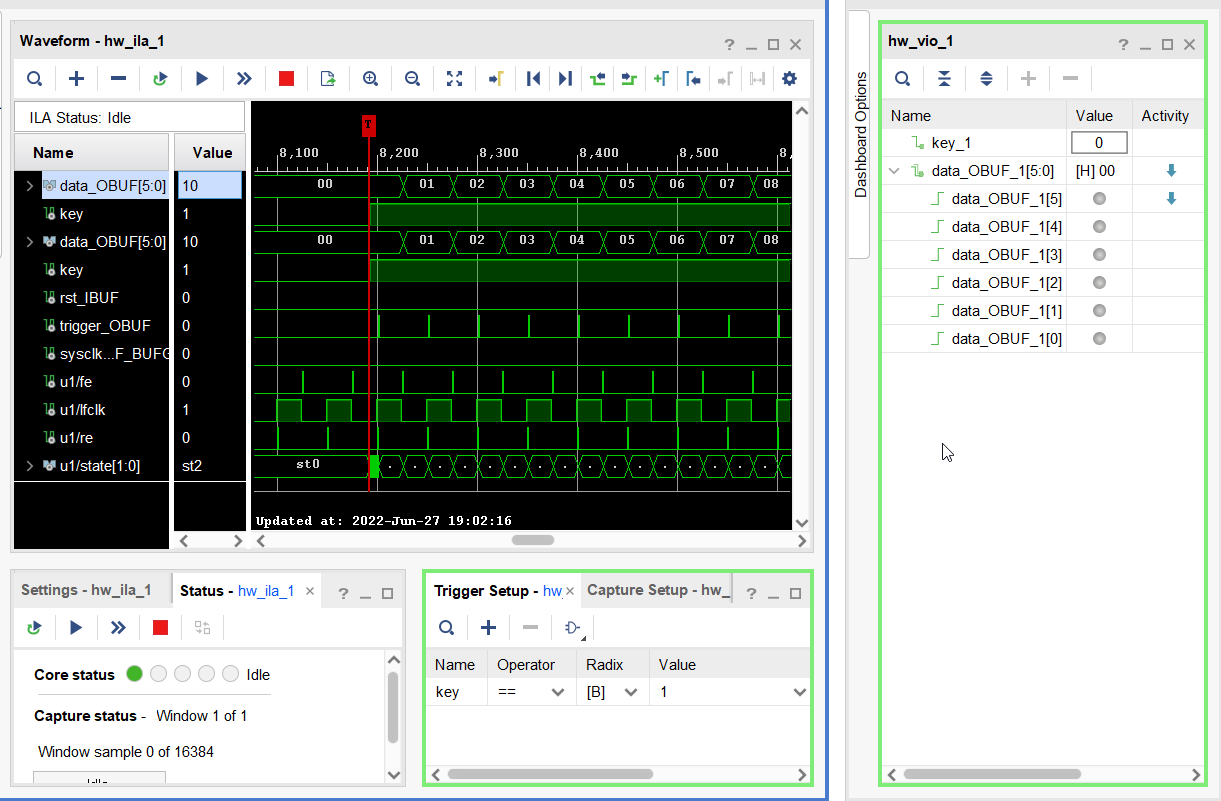
将需要查看的数据加入波形窗口，并在下方Trigger Setup页面加入key信号触发



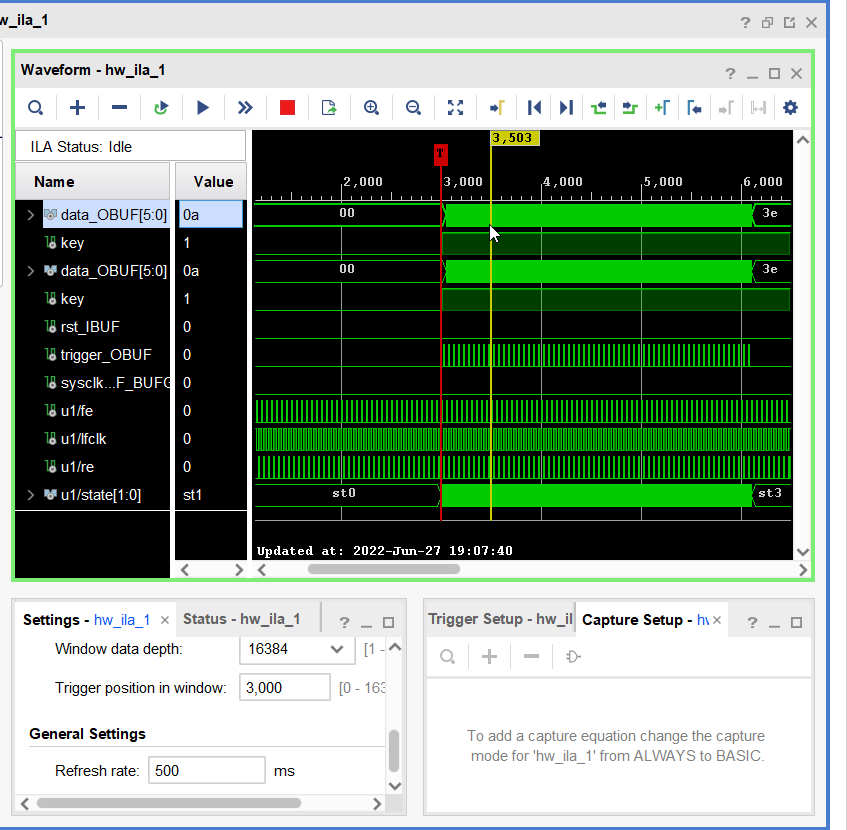
完成设置后进行调试，结果如下：



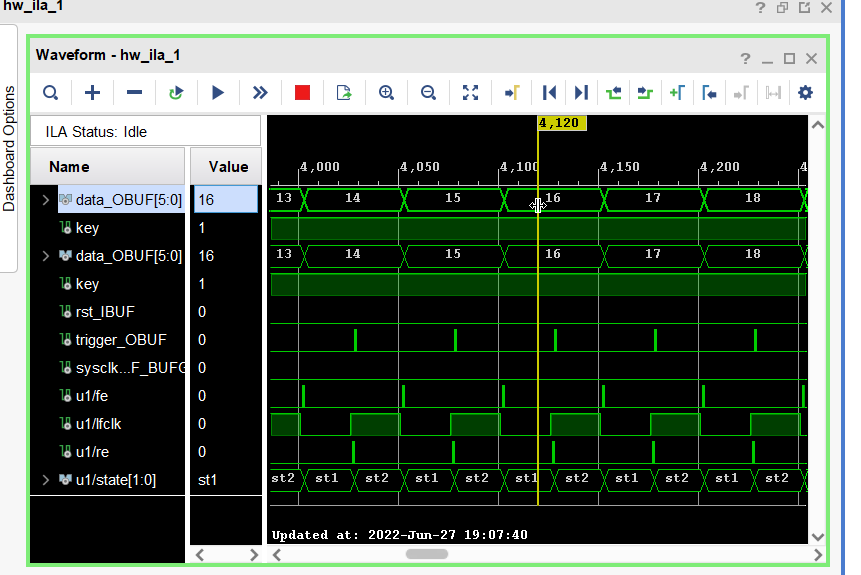
展开结果如下



将采样中点在窗口中显示位置设置为3000，得到波形如下：



部分波形如下所示：



【结果分析】

根据上面的调试记录，可以指导fsm模块是一个由外部触发的计数器，在按钮按下即key置为1时，会自动开始计数直到61。计数模式下，在时钟上升沿，状态机由st1变为st2；在下降沿，状态机由st2变为st1，完成计数后变为st3。状态机变化时钟由分频器分频后的系统时钟接管，divider模块对125MHz的系统时钟进行25分频得到5MHz时钟输送给fsm模块。

可以发现使用vio能够在软件层面完成硬件的调试，结合ila模块能够很好地对FPGA进行片上调试并观察到大部分所需信号。

与实验二中对RTC显示状态机使用的随机测试做比较，使用VIO和ILA调试可以更加直观地监控状态机变化，但是由于需要综合、布线后在片上测试，相对来说更耗费前期准备时间。对于逻辑不清晰且实现复杂的嵌套状态机更适合使用本方法；对于小规模的状态机，个人认为可以采用代码覆盖率高的随机测试进行快速模块验证。