**北京科技大学 计算机与通信工程学院**

**实 验 报 告**

实验名称： 数 字 逻 辑 实 验 一

学生姓名： 唐 誉 源

专 业： 计 算 机 科 学 与 工 程

班 级： 计 1503

学 号： 41503302

指导教师： 何 杰

实验成绩：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

实验地点： 机 电 楼 304

实验时间： 2017年3月1日

**一、实验目的与实验要求**

**1、实验目的**

本实验的目的是熟悉实验环境，学习如何使用 Vivado 2015.4 创建、综合、实 现、仿真等功能。

**2、实验要求**

1) 学习 Vivado-Design-Flow.pdf、约束文件.ppt，在实验报告中回答以下问题:

. a)  描述 Vivado 的设计流程

在跳转页中可以创建新项目或打开已有项目、打开例子项目和打开一些文档。在创建新项目时可以创建四种不同类型的项目，例如有RTL、综合后Post-synthesis、I/O规划和输入项目。在选择RTL级之后，加入资源文件、约束文件，选择FPGA。在定义项目的名称和位置之后，进行HDL语言的编写，并加入到RTL项目中，在Post-synthesized project creation那里选择合成后的网络表，加入约束文件，选择目标器件，索引原本存在的文件，或引入拷贝这些文件到项目中来。这样就创建了有HDL资源和综合设计的项目。

一般用Vivado创建RTL项目的基本流程是：配置项目源，IP集成，运行仿真，Open Elaborated Design按钮进行RTL分析，运行合成，Run Implement按钮运行实现，Program and Debug: 启动编程和调试。

同时，Vivado还可以进行约束文件管理、项目管理、项目设置、仿真设置等功能。

. b)  什么是网表

网表是设计的一种描述。包括了单元（设计目标，例如用户模块/设计内容的实例表达、库单元(BEL)的实例、硬件功能的通用技术表达、黑盒等）、引脚（单元的连结点）、端口（设计的顶层端口）和网络（引脚之间及引脚到端口的连结 ）。

. c)  什么是约束文件?通过 IO planning 完成的是什么方面的约束?

约束文件包括了管脚约束、区域约束和时序约束。 管脚约束：将模块的端口和FPGA的管脚对应；区域约束：将模块放置在FPGA的特点位置；时序约束：对数据建立、保持时间进行约束，保证设计在高速时钟下的工作可靠性等。

IO planning完成的是管脚约束。

. d)  Vivado 设计流程中，Synthesis 的作用是什么?

可以对工程进行综合，之后Open Synthesized Design 可以打开综合的结果（即在合成之后表达设计）。

. e)  Vivado 设计流程中，Implementation 的作用是什么?

运行实现，可以完成link, opt, power\_opt, place, phys\_opt和布线。Open Implemented Design 按钮可以装载实现设计。

2) 在实验报告中提交上述实验内容的 Verilog 代码、仿真结果图、RTL详细设 计 图 ( 参 考 Vivado-Design-Flow.pdf 第 11 页 Open Elaborated Design—>Schematic)、综合实现图(参考Vivado-Design-Flow.pdf 第13页

Open Synthesized Design—>Schematic)以及实验现象图(照片); 3) 提交实验报告和完整的工程文件。

**二、实验设备（环境）及要求**

实验室提供Windows 7操作系统下的Xilinx开发平台vivado2015.4环境，以及EGO开发板。

**三、实验内容与步骤**

**1、实验1**

（1）实验内容

* 学习视频，了解 Vivado 设计流程和功能:”EGo 五分钟快速上手.mp4” 和“Ego 五分钟搭建你的数字积木.mp4”。
* 按照“EGo 五分钟快速上手——流水灯.pdf”完成，流水灯实验;

（2）主要步骤

观看“实验1.1”文件夹中“EGo五分钟快速上手”视频，熟悉Vivado设计流程，再根据“EGo五分钟快速上手——流水灯.pdf”PDF 文档做出实验结果;

利用vivado设计完成流水灯的主要步骤为：

* 在 Vivado 中创建RTL设计
* 进行HDL语言编写

**flowing\_light.v**

`timescale 1ns / 1ps

module flowing\_light( input clk,

input rst,

output [15:0] led);

reg [23 : 0] cnt\_reg; reg [15 : 0] light\_reg;

always @ (posedge clk) begin

if (rst)

cnt\_reg <= 0;

else

cnt\_reg <= cnt\_reg + 1;

end

always @ (posedge clk) begin

if (rst)

light\_reg <= 16'h0001;

else if (cnt\_reg == 24'hffffff) begin

if (light\_reg == 16'h8000) light\_reg <= 16'h0001;

else light\_reg <= light\_reg << 1;

end end

assign led = light\_reg; endmodule

* 设置激励仿真

**test**：

`timescale 1ns / 1ps

module test( );

reg clk;

reg rst;

wire [3 : 0] led;

flowing\_light u0( .clk(clk),

.rst(rst), .led(led) );

parameter PERIOD = 10;

always begin clk = 1'b0;

#(PERIOD/2) clk = 1'b1; #(PERIOD/2);

end

initial begin clk = 1'b0; rst = 1'b0; #100;

rst = 1'b1;

#100; rst = 1'b0;

end endmodule

* 建立仿真
* 通过I/O planing添加管脚设计
* 通过编写约束文件添加管脚约束

Verilog代码：

**top\_xdc.xdc**

set\_property PACKAGE\_PIN K3 [get\_ports {led[0]}]

set\_property PACKAGE\_PIN M1 [get\_ports {led[1]}]

set\_property PACKAGE\_PIN L1 [get\_ports {led[2]}]

set\_property PACKAGE\_PIN K6 [get\_ports {led[3]}]

set\_property PACKAGE\_PIN J5 [get\_ports {led[4]}]

set\_property PACKAGE\_PIN H5 [get\_ports {led[5]}]

set\_property PACKAGE\_PIN H6 [get\_ports {led[6]}]

set\_property PACKAGE\_PIN K1 [get\_ports {led[7]}]

set\_property PACKAGE\_PIN K2 [get\_ports {led[8]}]

set\_property PACKAGE\_PIN J2 [get\_ports {led[9]}]

set\_property PACKAGE\_PIN J3 [get\_ports {led[10]}]

set\_property PACKAGE\_PIN H4 [get\_ports {led[11]}]

set\_property PACKAGE\_PIN J4 [get\_ports {led[12]}]

set\_property PACKAGE\_PIN G3 [get\_ports {led[13]}]

set\_property PACKAGE\_PIN G4 [get\_ports {led[14]}]

set\_property PACKAGE\_PIN F6 [get\_ports {led[15]}]

set\_property PACKAGE\_PIN R15 [get\_ports rst]

set\_property PACKAGE\_PIN P17 [get\_ports clk]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[15]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[14]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[13]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[12]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[11]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[10]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[9]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[8]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[7]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {led[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports clk]

set\_property IOSTANDARD LVCMOS33 [get\_ports rst]

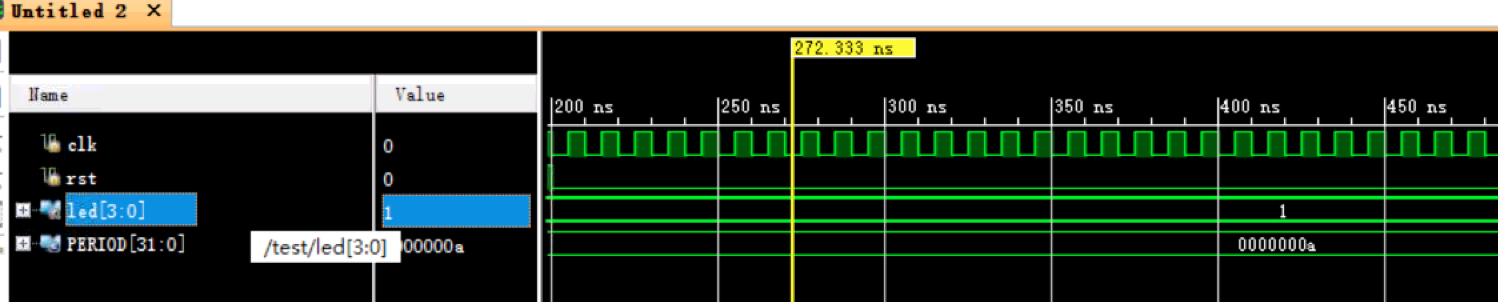
create\_clock -period 8.000 -name clk\_pin -waveform {0.000 4.000} -add [get\_ports clk]

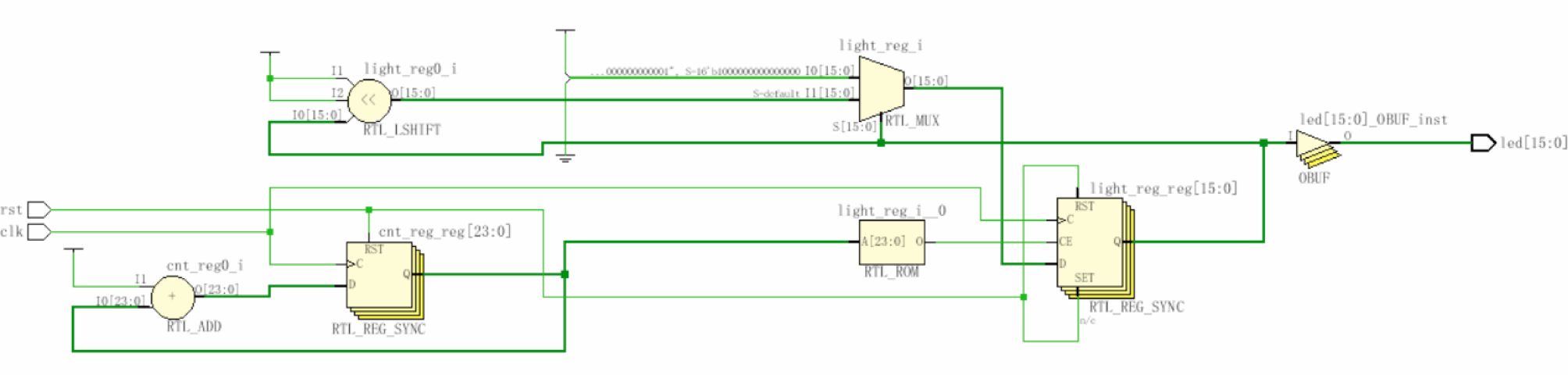
set\_input\_delay -clock [get\_clocks -filter { NAME =~ "\*" }] -add\_delay 0.000 [get\_ports rst]

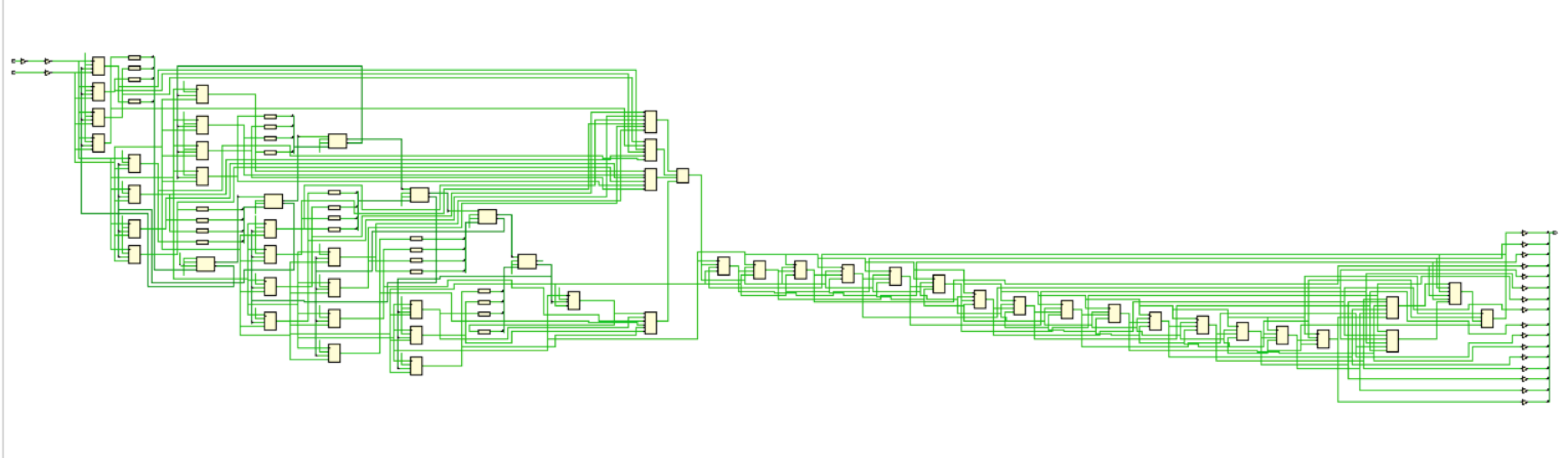
set\_input\_delay -clock [get\_clocks -filter { NAME =~ "\*" }] -min -add\_delay -0.500 [get\_ports rst]

set\_output\_delay -clock [get\_clocks -filter { NAME =~ "\*" }] -add\_delay 0.000 [get\_ports -filter { NAME =~ "\*" && DIRECTION == "OUT" }]

* 添加时序约束
* 生成 Bitstream 文件 下载到 FPGA开发板中

**四：实验结果与分析**

仿真结果图

流水灯RTL级原理图

综合实现图

由Vivado的可视化特性，我们得以观察到我们由HDL语言编程的流水灯的仿真时序关系，并在网络表、电路图和RTL设计图中交叉对比，调试完善流水灯程序。

根据Vivado的仿真结果图、流水灯RTL级原理图和综合实现图可看出成功根据“EGo 五分钟快速上手——流水灯.pdf”完成流水灯实验，并在EGO电路板上产生相应响应。



实验现象图

**五：结论（讨论）**

**1、实验结论**

本次试验熟悉了Xilinx开发平台vivado2015.4环境，以及EGO开发板，学习到了如何使用Vivado创建、综合、实现、仿真等功能。

**2、讨论**

Vivado IDE工具有诸多好处，他使FPGA的性能可以预计。从上到下的设计方法能加速实现目标的设计、分析功能，使我们的脚本创建更加容易。使用共同的约束语言(XDC)使大多数设计过程只需要按下按钮。且在整个FPGA设计过程中使用共同的数据模型，是我们在运行时间和存储资源上受益 。

**六、教师评审**

|  |  |
| --- | --- |
| **教师评语** | **实验成绩** |
| 签名：  日期： |  |