东南大学

《数字逻辑与计算机体系结构（含实验）》

实验报告

实验四　数字模块设计与验证

姓　　名：赵舞穹　　**学　　号**：61520522

同　　组：郑瑞琪　　**学　　号**：61520523

专　　业：工科试验班　　**实 验 室**：计算机硬件技术

实验时间：2021 年 11 月 19 日　　**报告时间**：2021 年 11 月 29 日

评定成绩：　　**评阅教师**：冯熳

目录

一. 实验目的

1. 通过实验学习理解和掌握虚拟器件/接口、IP 和基于平台的数字系统设计方法，形成软核和硬核、宏单元、虚拟器件、IP 设计及验证的概念和基本的调用构建过程；

2. 通过实践加深对可编程数字系统模块概念的正确理解，进一步熟悉掌握 FPGA 数字模块的仿真验证方法和开发设计过程及 FPGA 硬件演示应用方法，初步理解掌握 Xilinx 的 IP 核和 Altera 的参数化宏功能模块库（Library of Parameterized Modules–LPM）的原理，分析对比计算机典型数字元件模块功能设计和典型应用方法，总结形成计算机系统与接口相关模块可能的 HDL 实现方法。

二. 实验内容

(一) XILINX Vivado 加减法器的实现

1. Verilog 模块调用 IP 核搭建

首先如图 1 配置好一个 `Adder/Subtracter` 的 IP 核，这样就相当于完成了整个模块而没有具体的写出代码.

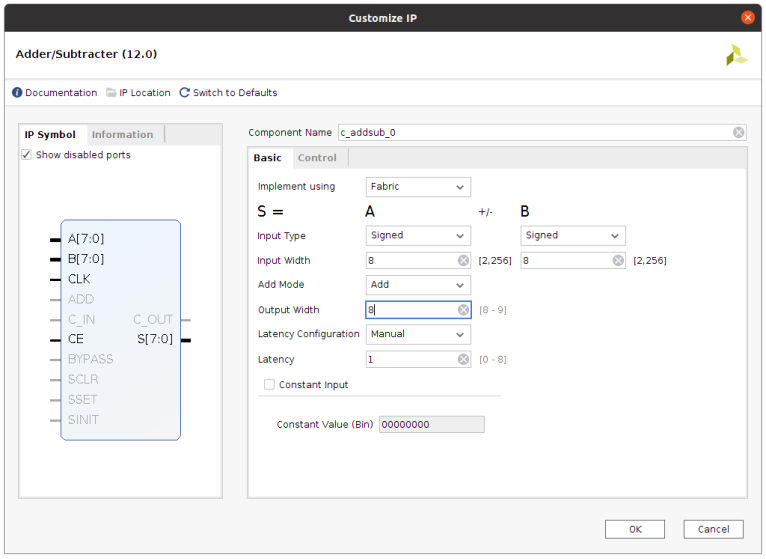

图 1: Vivado 中设置 8 位 `Adder/Subtracter` 模块

随后需要添加 testbench 代码，此处借鉴老师提供的 `demo.v`[1].

```
module tvj_demo( // use my symbol 'tvj'
  input t1,
  output t2
  );
  reg [7:0] Am,Bm;
  wire [7:0] Sm;
```

```
  reg CLKm=0,CEm=0;
  c_addsub_1 myadder (
  .A(Am),      // input wire [7 : 0] A
  .B(Bm),      // input wire [7 : 0] B
  .CLK(CLKm), // input wire CLK
  .CE(CEm),    // input wire CE
  .S(Sm)       // output wire [7 : 0] S
  );
  always #2 CLKm = ~CLKm; // clock signal
  initial begin
    #2;
    CEm = 1;
    Am = 38;
    Bm = 88;
    #6;
    Am = 55;
    Bm = -24;
    #5;
    Am = -70;
    Bm = -38;
    #5;
    CEm = 0;
    Am = 8'b1011001;
    Bm = 8'b11010001;
    #20;
  end
endmodule
```

进行 Behavioral Simulation，得到的行为仿真图如图 2 所示.

结果分析 1: 行为仿真结果

行为仿真结果显示它成功进行了 8 位有符号数（即按照补码编码）的和差运算.

分别完成 RTL，Synthesis，Implementation 的仿真，分别得到图 3，5，6 的结果.

结果分析 2: RTL，Synthesis，Implementation 的仿真

RTL 级仿真是寄存器级的，而我们将其展开，如图 4 所示，是嵌套的具有超前进位的结构并不需要使用到寄存器. Synthesis（综合）则需要将更多的内容考虑到了，例如具体的输入输出始时钟上存在一定的差异. Implementation（实现）需要考虑到具体开发板的情况，我在创建项目的时候是按照默认选择的. Implementation 会完成 Translation，Mapping，Place&Route 的工作，这些是在 Synthesis 中没有的. 图 6 结构如此简单我认为可能是对应的板子本身就具有 `Adder/Subtracter`.

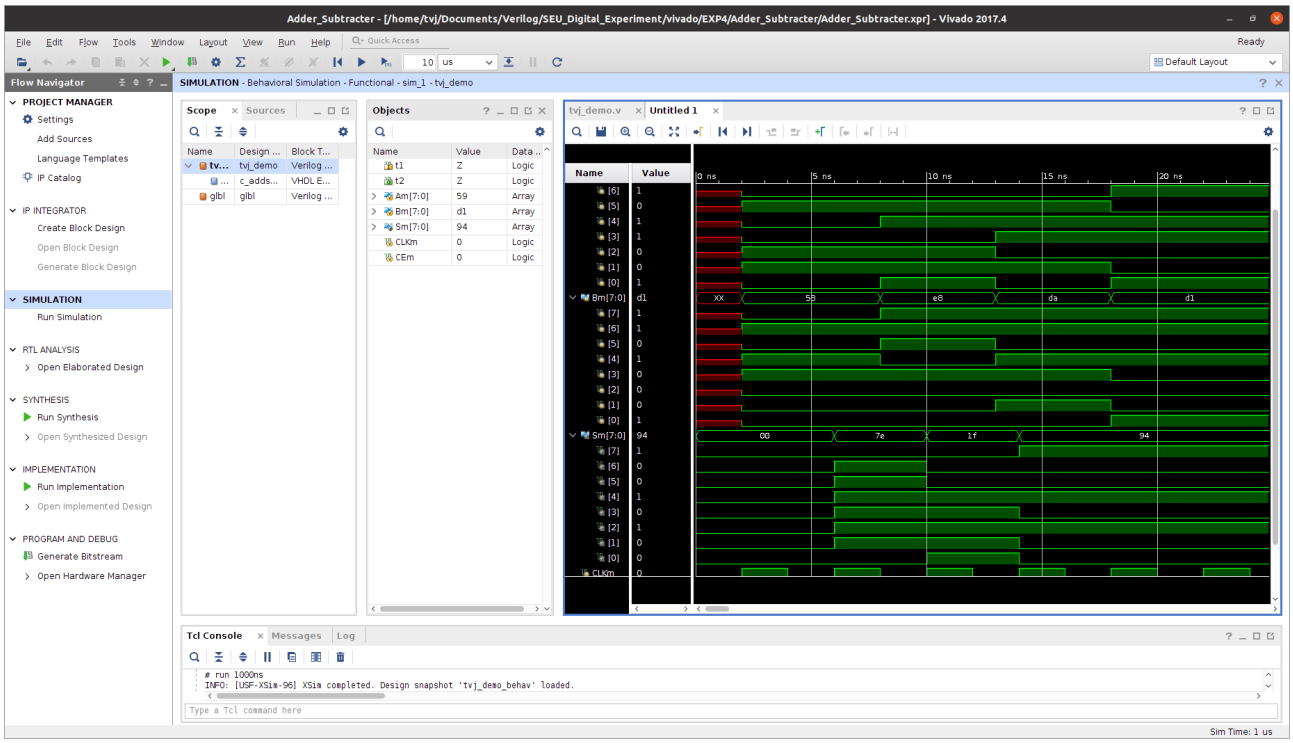

图 2: 行为仿真图

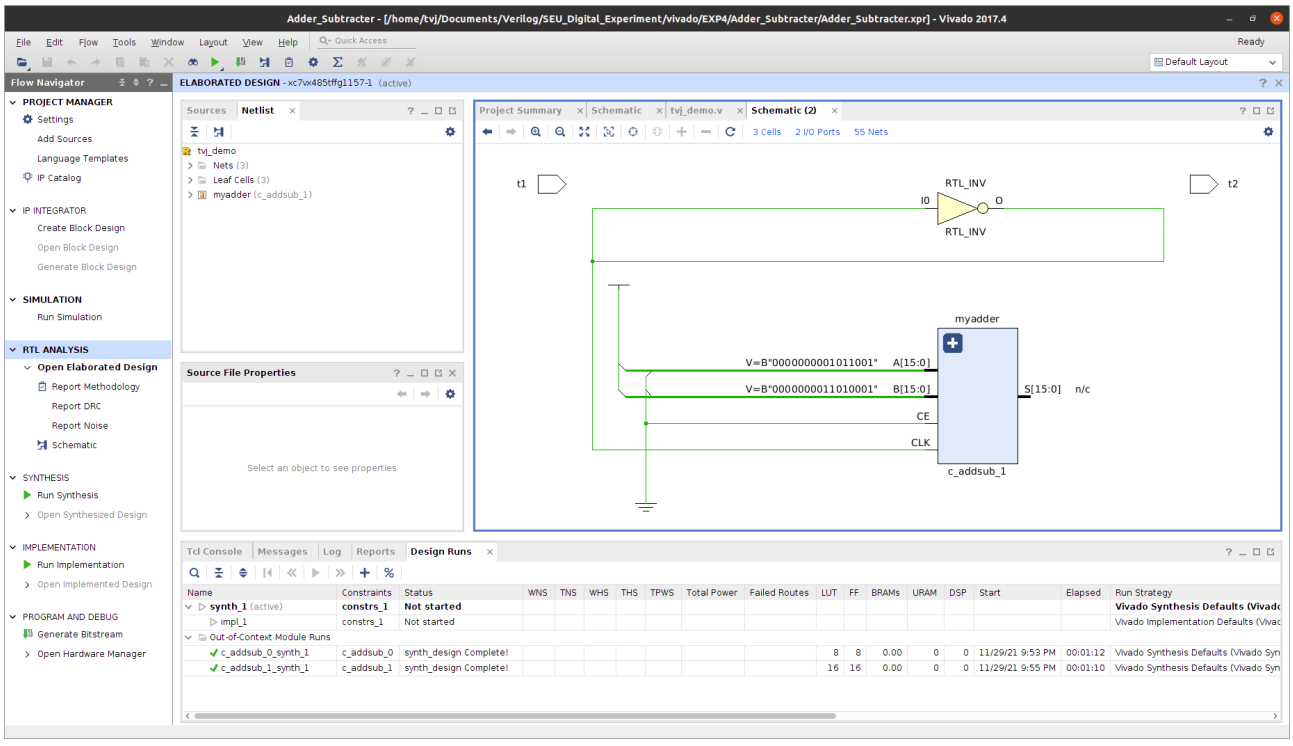

图 3: RTL 级电路

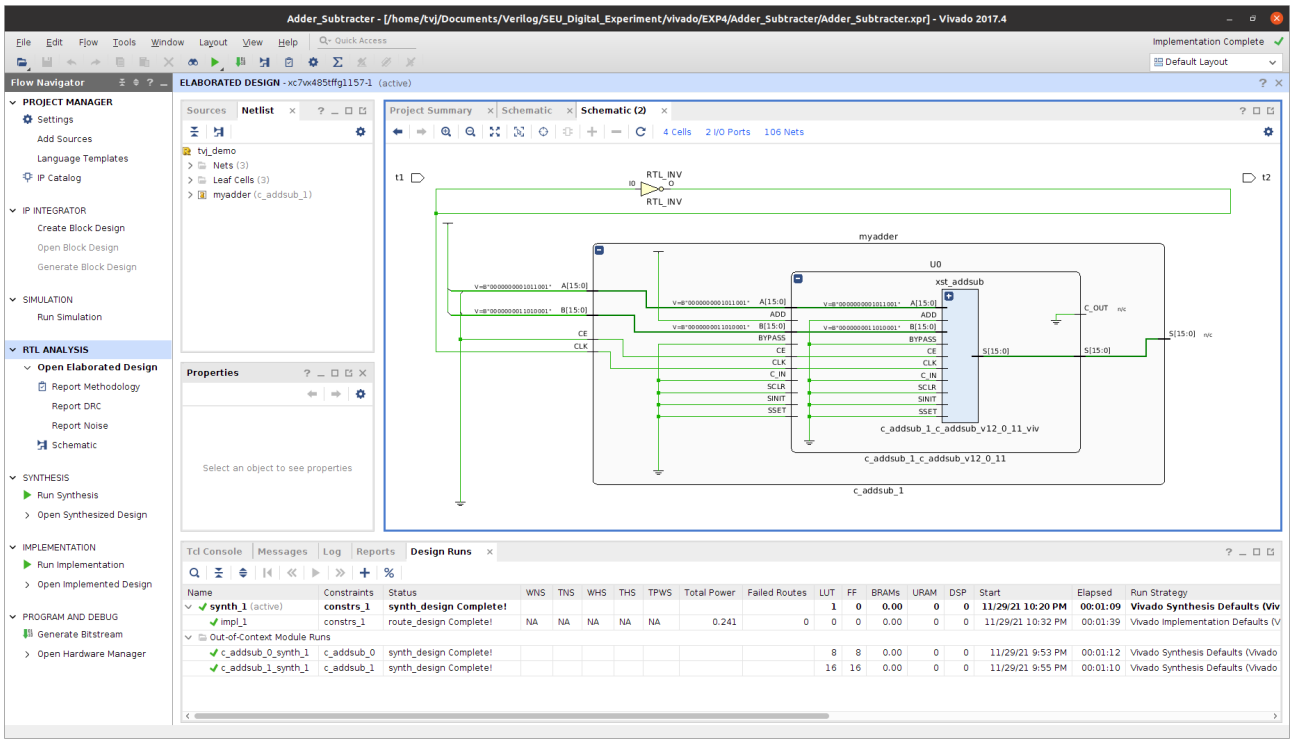

图 4: RTL 级电路（展开两级）

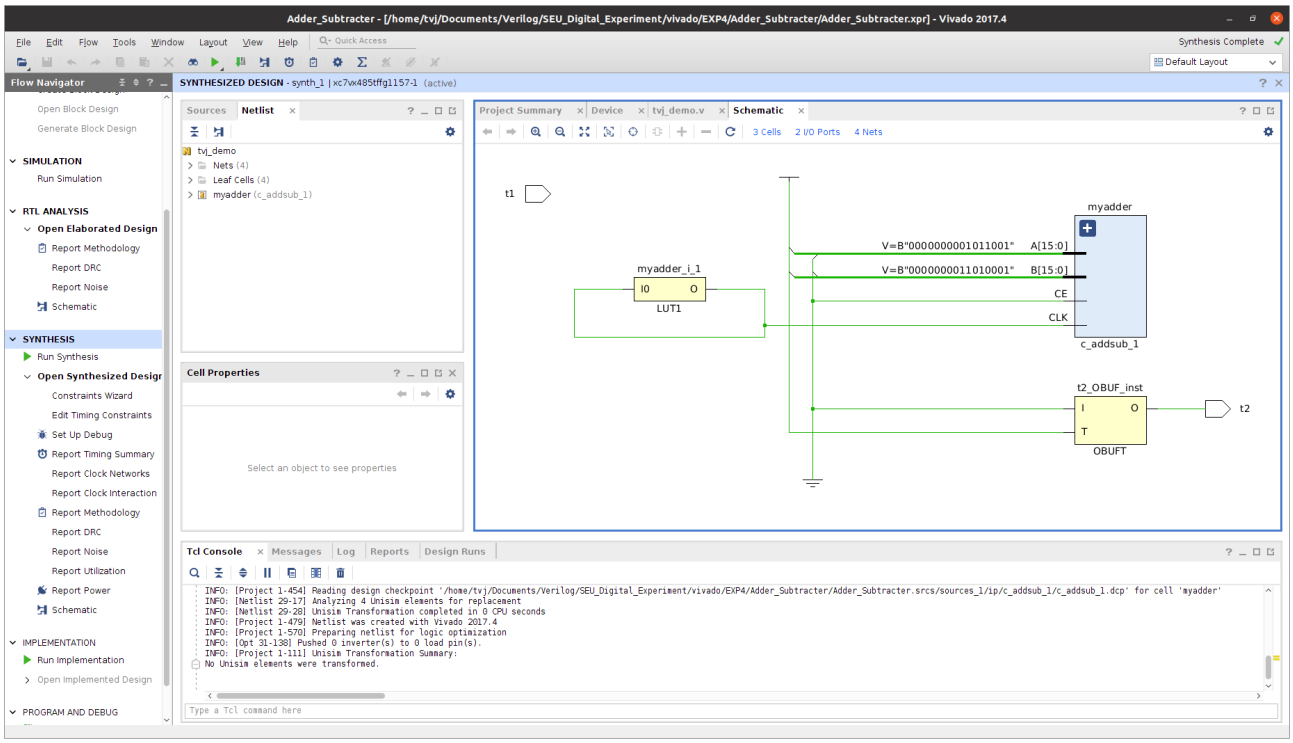

图 5: 综合电路

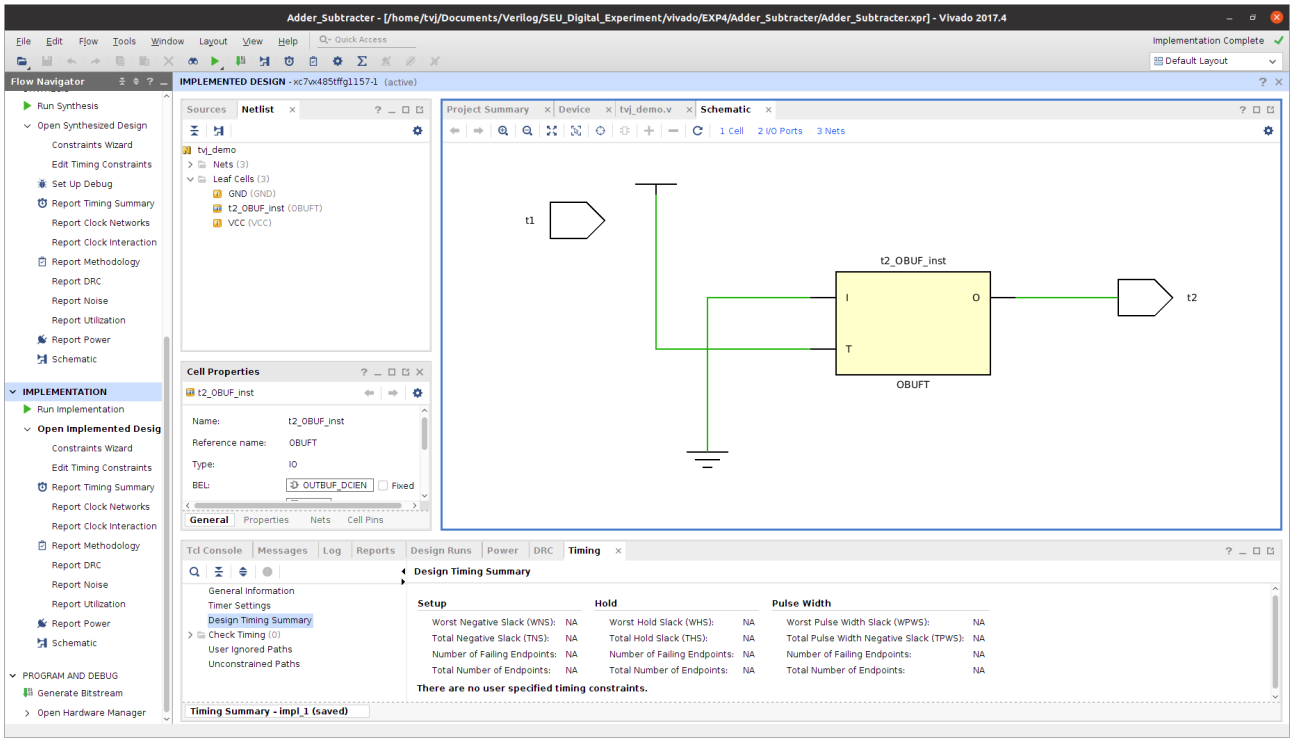

图 6: 实现电路

2. （原理）框图设计（Block design）调用 IP 核搭建

使用，Block Design，设计输入和输出的 port，并用鼠标连接对应的端口. 绘图结果如图 7 所示.

注意 1

此处我没有按照实验指导书做 8 位的，而是 15 位，因此都是 `[14:0]`.

得到的 RTL 级电路如图 8 所示，生成的 wrapper 代码如图 9 所示.

结果分析 3: 框图设计（Block design）调用 IP 核搭建

使用框图设计整体思路是比较清晰的，选择 IP 核之后设置对应的输入输出端口完成链接. 生成出的 RTL 级电路看上去就结构清晰，此外生成的 wrapper 代码也非常易读，与使用框图设计出来的结果一致. 这个过程其实很像软件制作的 Qt，UI 界面既可以使用代码直接完成，又可以通过拖拽的方法设置 `.ui` 文件以实现快速的设置.

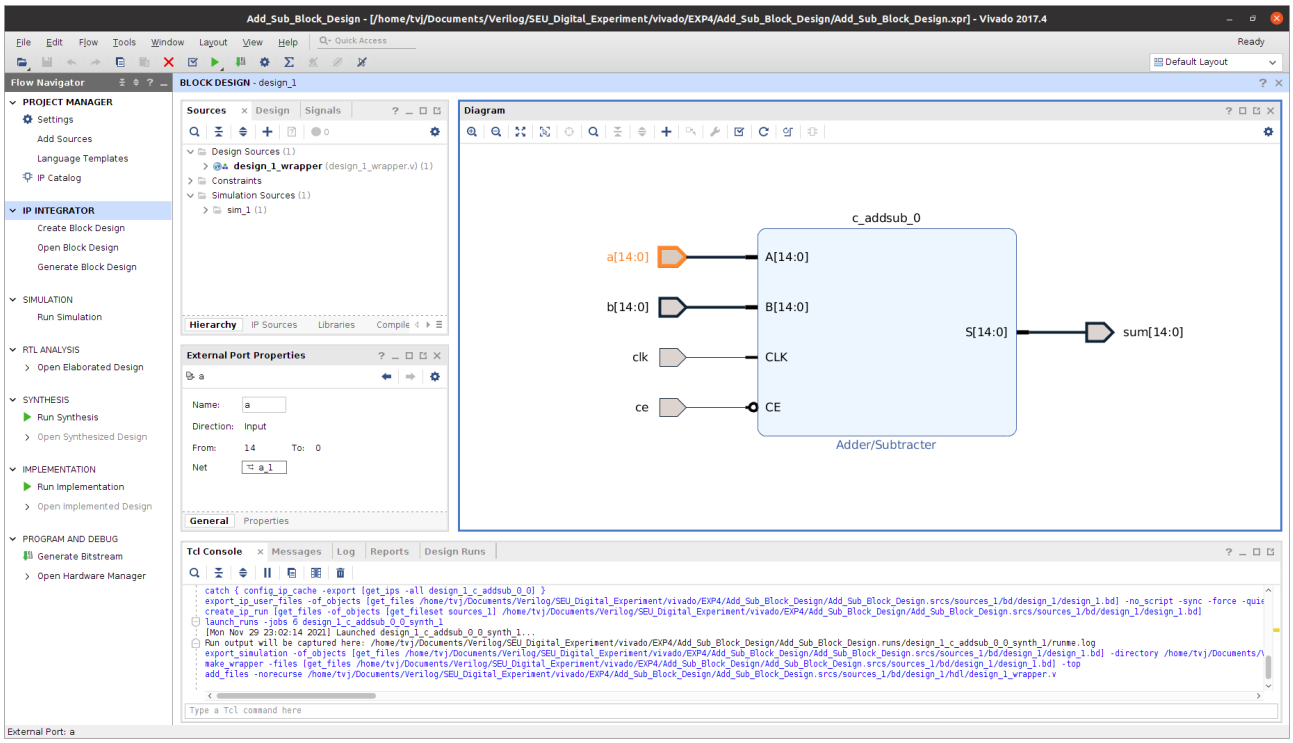

图 7: Block Design 界面

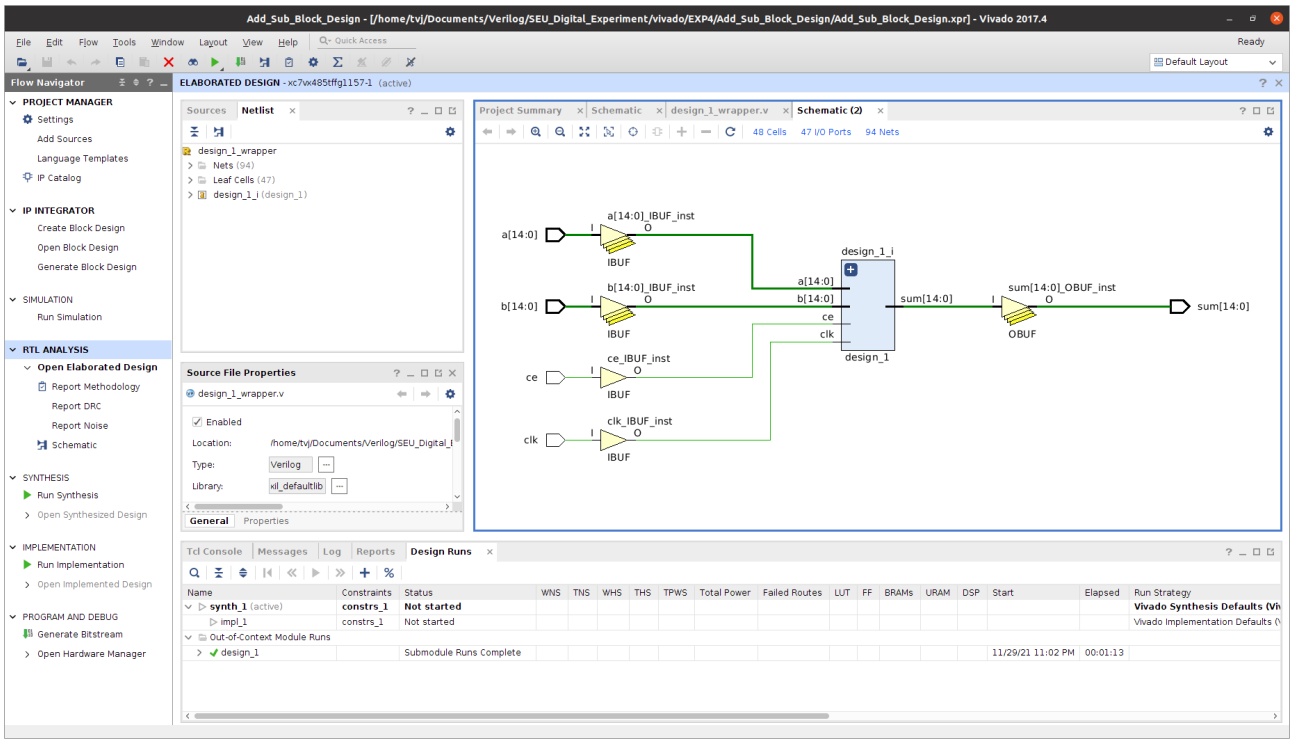

图 8: Block Design 的 RTL 级电路

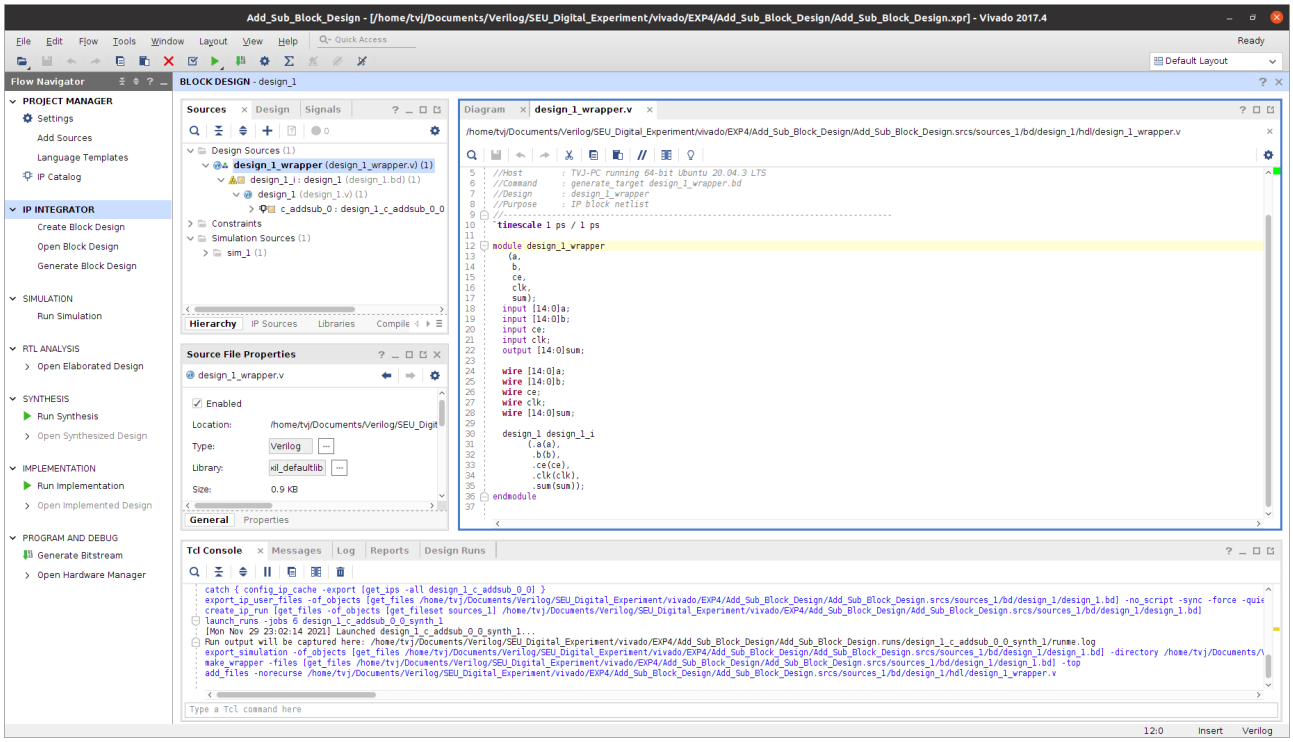

图 9: Block Design 的 wrapper 代码

(二) 基于 dff 的 4 位移位寄存器

带异步清零的 D 触发器 dff 之前已经实现过了，此处代码不单独拿出来，图 10 是其 RTL 级电路图.

4 位移位寄存器的核心实现代码如下.

```
module ShiftRegister(C, L, RTL, R, D, Q, nQ);
  parameter n = 4;
  reg [n - 1:0] Dm;
  input wire C, L, RTL, R;
  input wire [n - 1:0] D;
  output wire [n - 1:0] Q;
  output wire [n - 1:0] nQ;

  always@(posedge C or posedge L)
  begin
    if(R)
      assign Dm = 0;
    else if (L == 0)
      begin
        if (RTL == 1)
          assign Dm = {Q[n - 2:0], D[0]}; // shift right
```

```
17          else
18            assign Dm = {D[n - 1], Q[n - 1:1]};
19        end
20      else
21        begin
22          assign Dm = {D[n - 1:0]};
23        end
24    end
25
26    genvar i;
27    generate for (i = 0; i < n; i = i + 1)
28      DFlipFlop dff(.C(C), .D(Dm[i]), .Q(Q[i]), .nQ(nQ[i]));
29    endgenerate
30  endmodule
```

结果分析 4

实现 IP 核封装的方式类似上一个实验内容，总体感觉在使用上区别不大，都是较为自然的，正如 C++ 一样进行一个较为完成的类封装.

4 位移位寄存器代码参考：https://github.com/AliRezaBeigy/ShiftRegister.

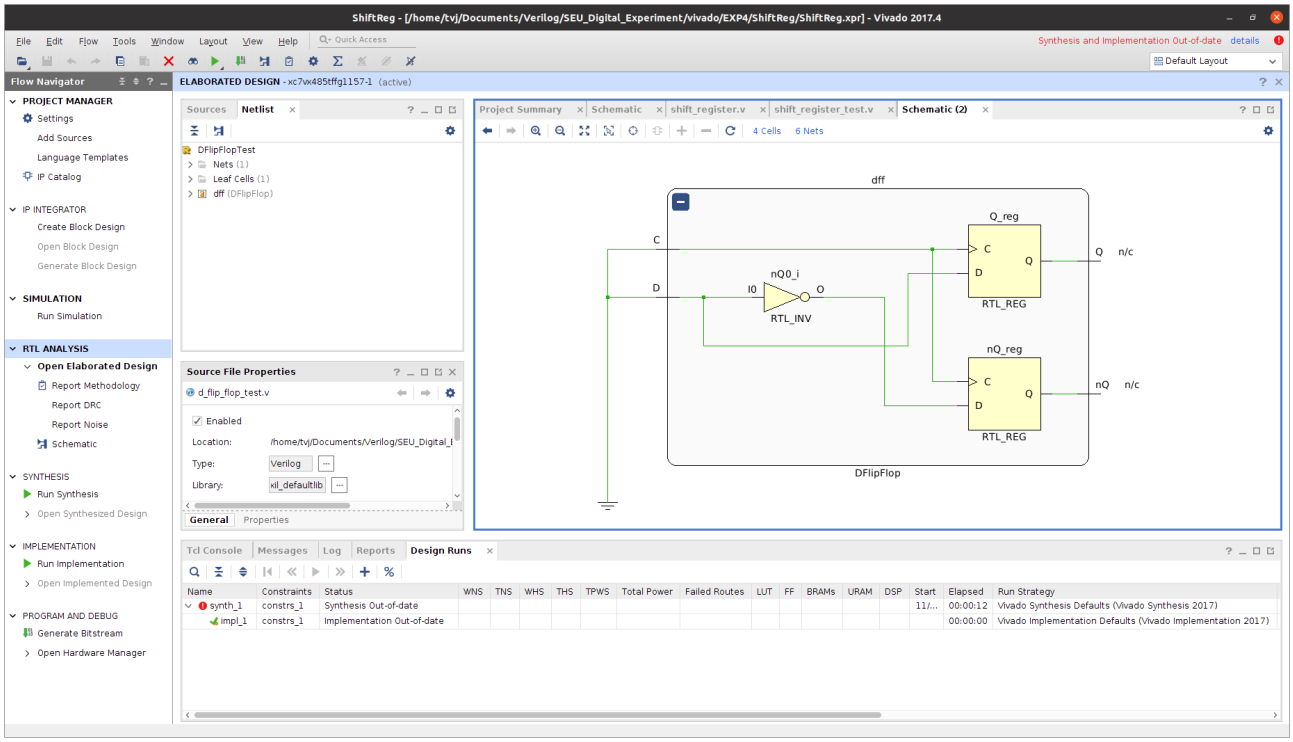

图 10: 4 位移位寄存器的行为仿真图

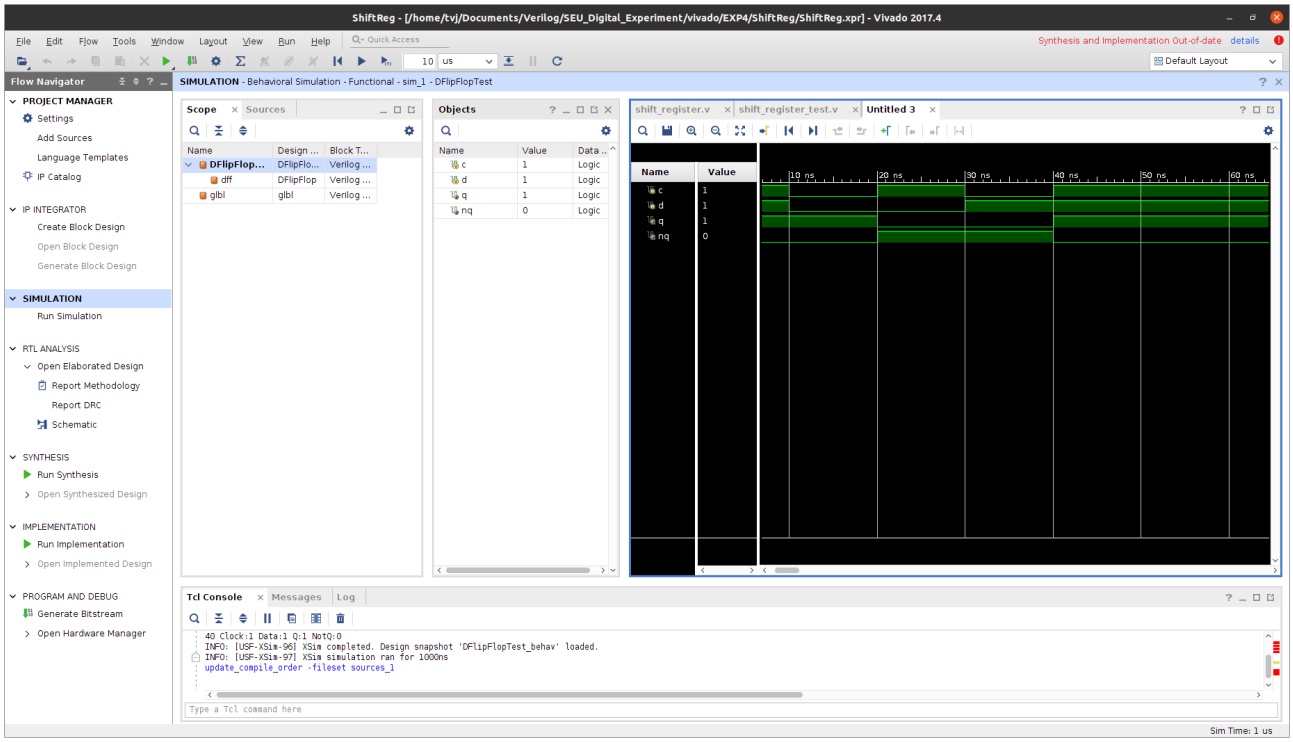

图 11: 4 位移位寄存器的行为仿真图

(三) Altera 的使用

首先，搭建图 12 的电路图，行为仿真得到图 13 的结果，验证了累加器的工作方式. 接下来需要配置引脚文件，由于实验指导书[1] 附带的代码没有提供银角文件，我们需要自己设计，这个过程[1] 上也没有介绍，我们通过学习网上资料，完成了如下的工作. 首先，需要如图 14 选择我们使用的开发板型号：`Cyclone EP1C6Q240C8`，再设置每个引脚的连接（如图 15），这个过程会十分繁琐，设置好部分引脚的结果如图 16 所示，右边的引脚伤未完成分配. 如果引脚是已经写好的，可以用图 17 的方法导入（但是很不幸，提供的材料没有，选择的文件都是我们自己刚新建的那一个）.

结果分析 5

- 图 13 中可以看到进位信息（不过在屏幕上一动起来效果会更好一些），对于累加器的理解有了更深入的理解；
- 通过 LD/CNT 控制 `A7..0` 可以在每个时钟周期自加一或者接受来自数据总线的数据；
- 由于引脚分配过于困难，我们在后面采用了提供的程序来做烧录板子的工作.

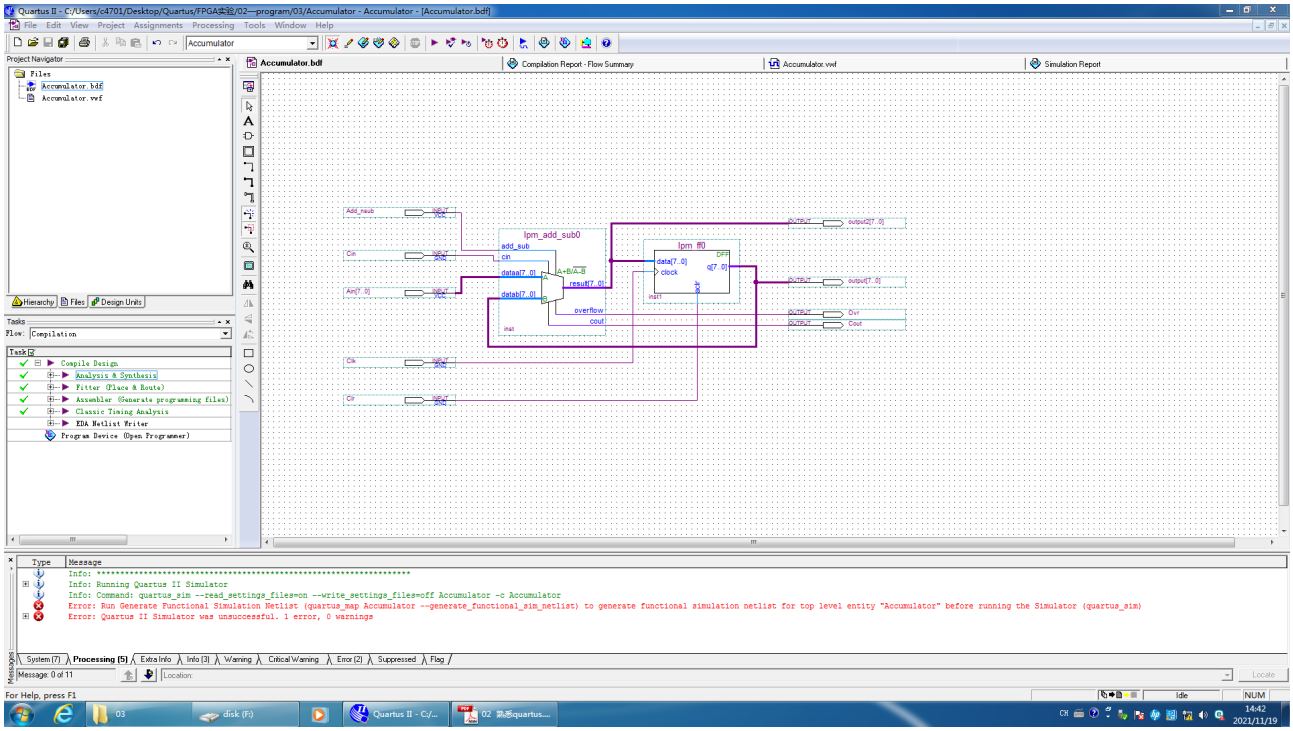

图 12: 基于 LPM 库的累加器电路原理图

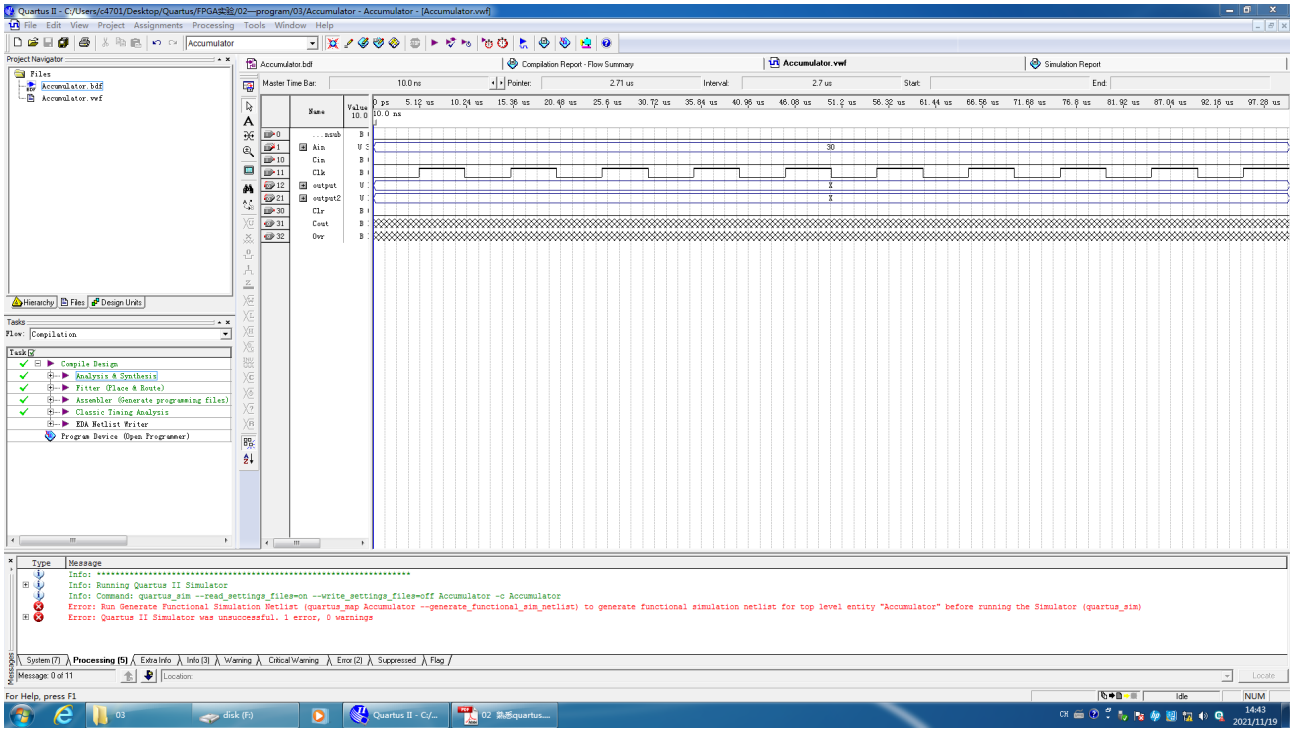

图 13: 仿真结果

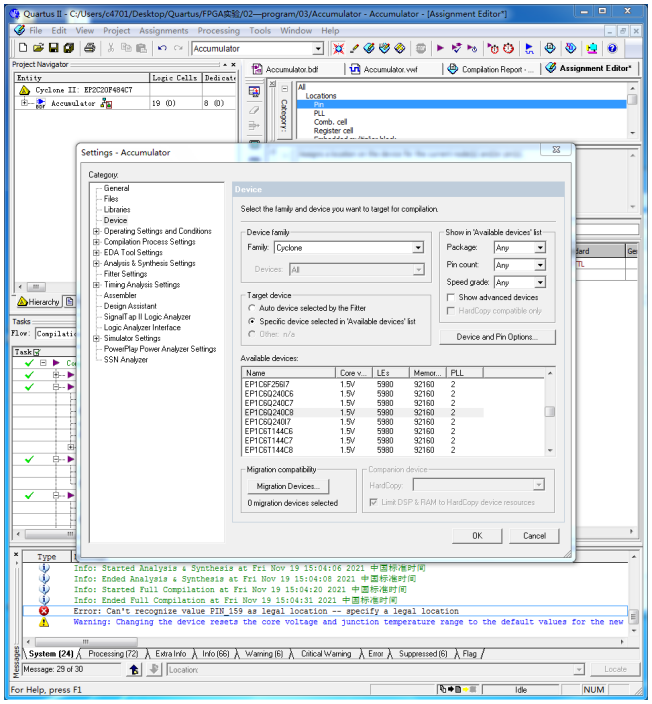

图 14: 选择仪器型号

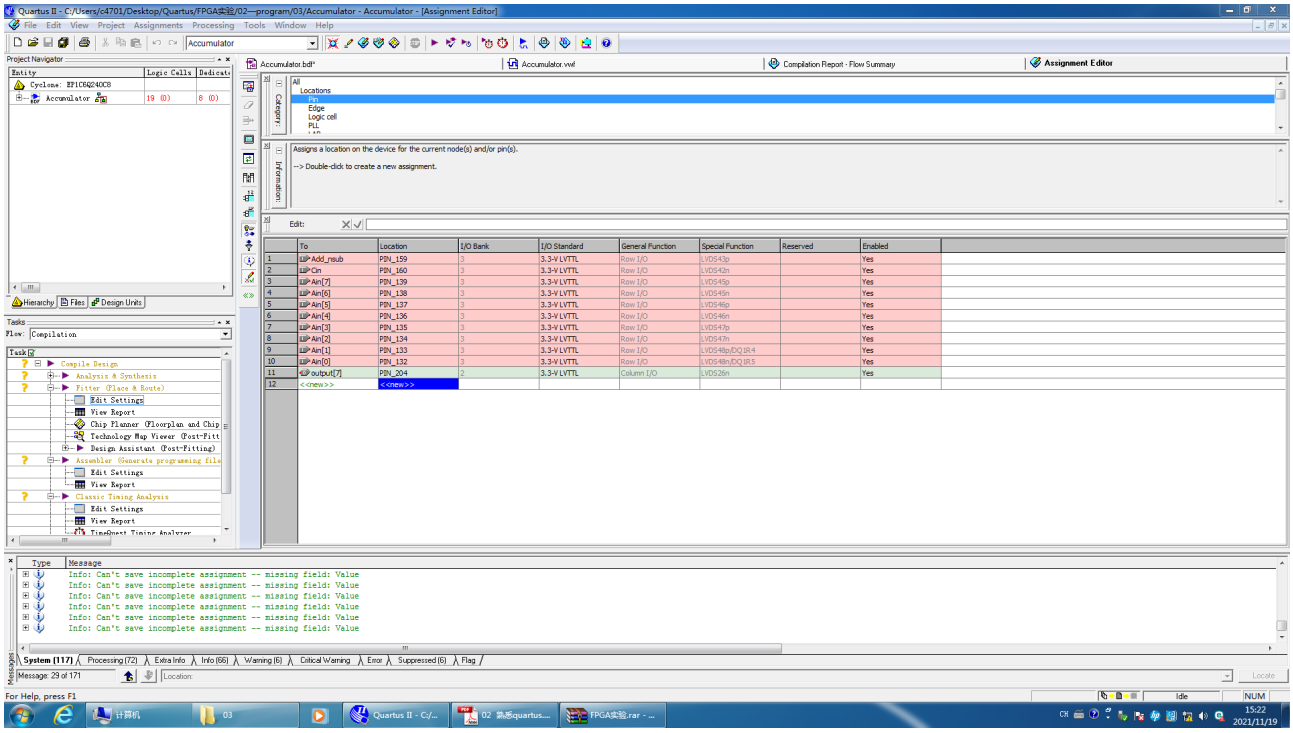

图 15: 设置引脚

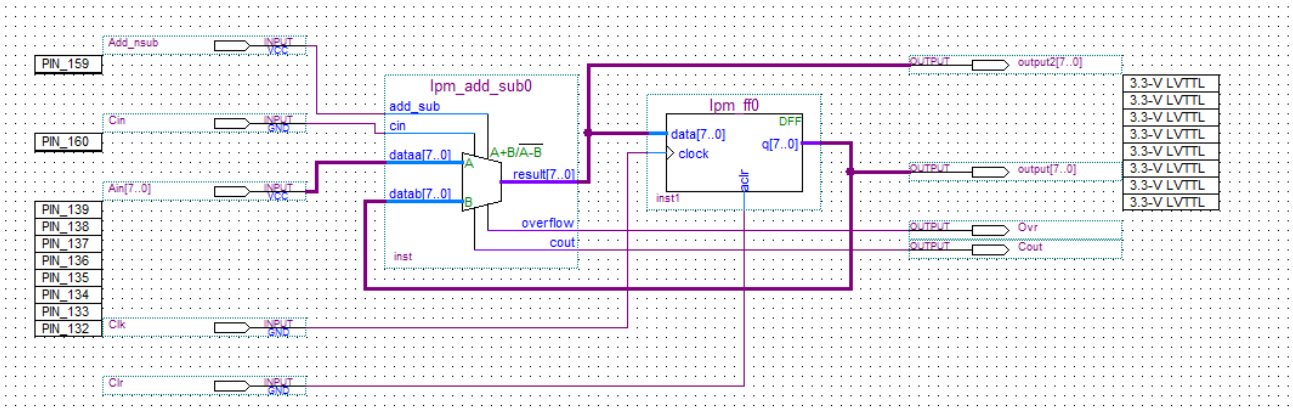

图 16: 设置引脚的结果

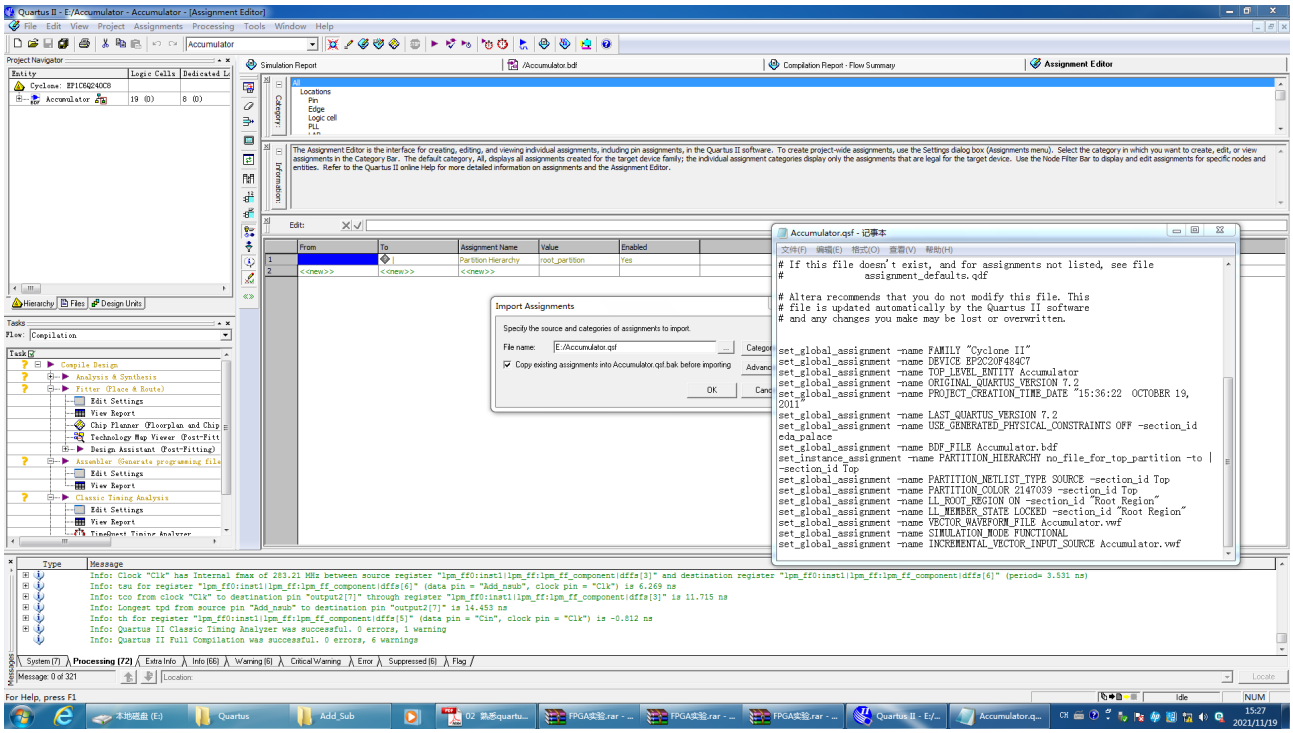

图 17: 导入引脚配置文件

我们以提供的项目为蓝本，设计了 74138 和灯测试及测试单元的电路并进行了测试，图 18 是 74138 电路设计图，图 19 是灯测试的电路，图 20 是测试模块的电路. 以图 21 的方式将项目烧录至开发板中，实现了预计的效果，主要是对整个开发的流程有了更加深入的了解，最后图 22 是实现的效果.

结果分析 6

整个过程较为复杂且中间的许多步骤都需要较为仔细的学习，而这次实验时间有限，主要目的是做一个大致的了解，对与整个开发流程有一个印象.

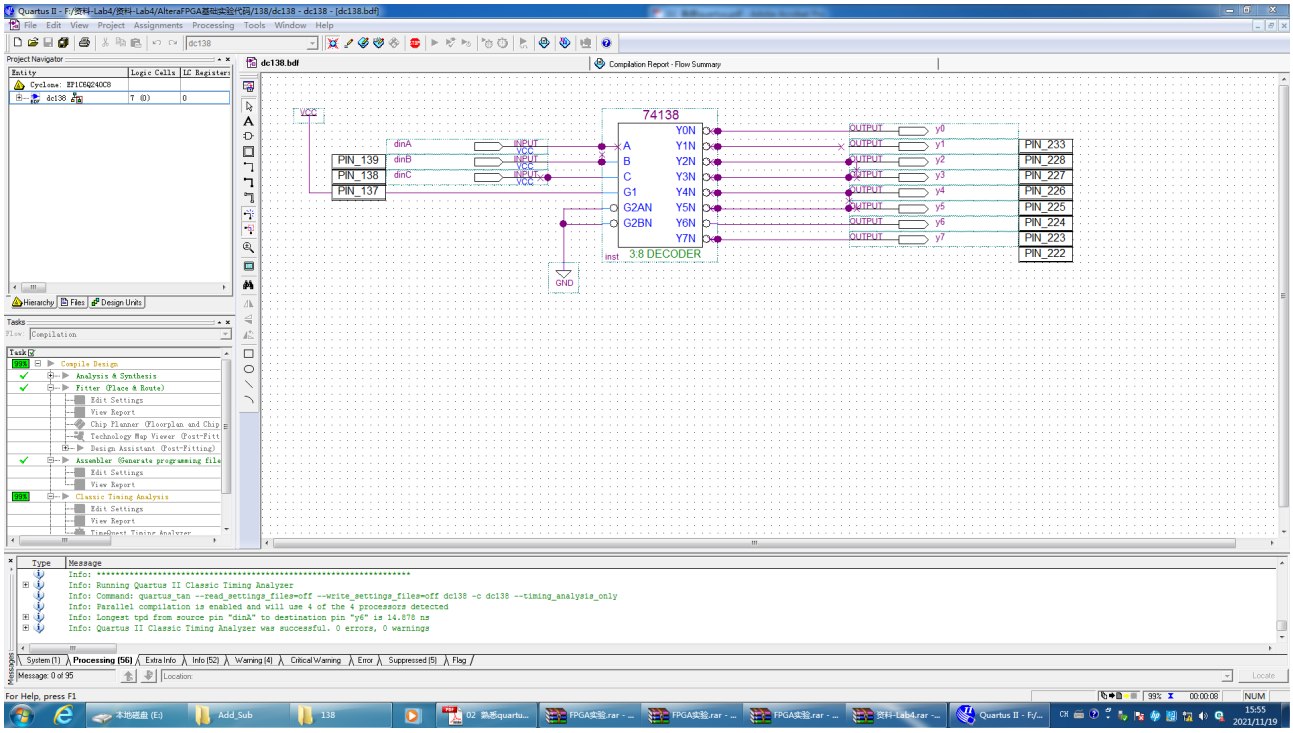

图 18: 74138 电路设计

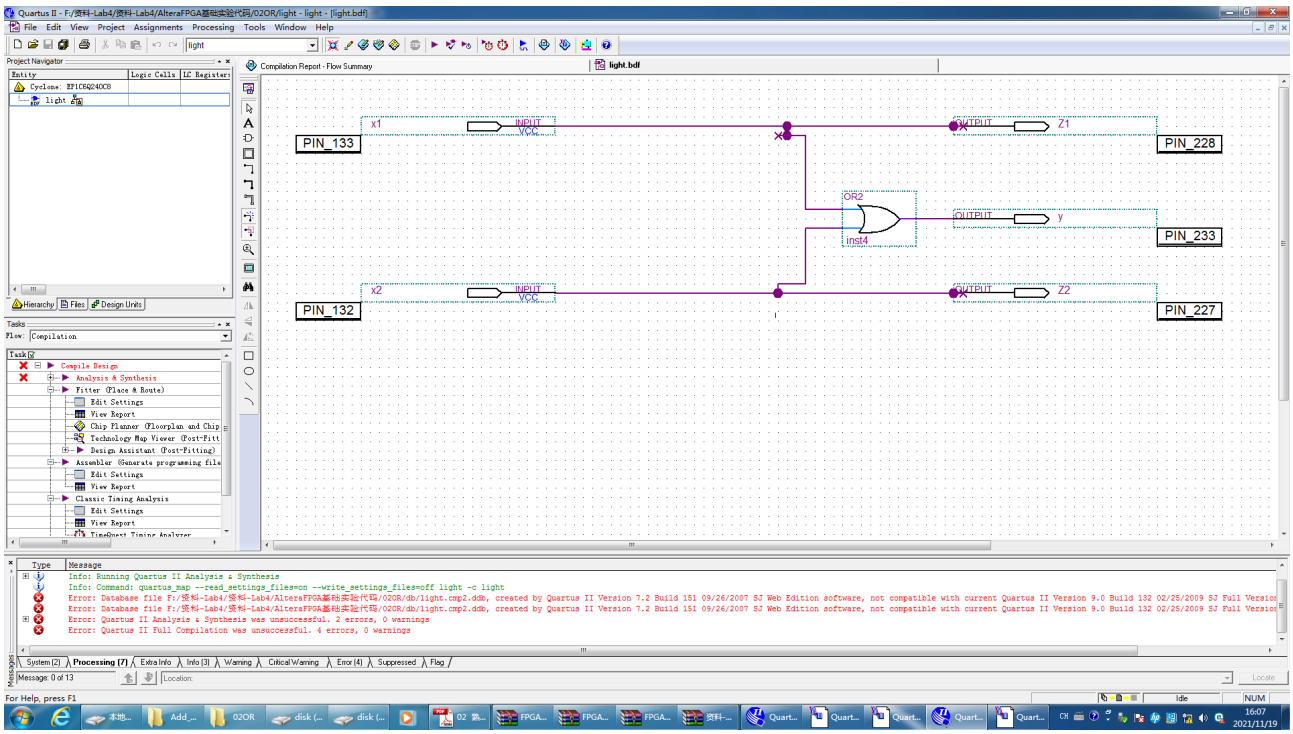

图 19: 灯测试电路设计

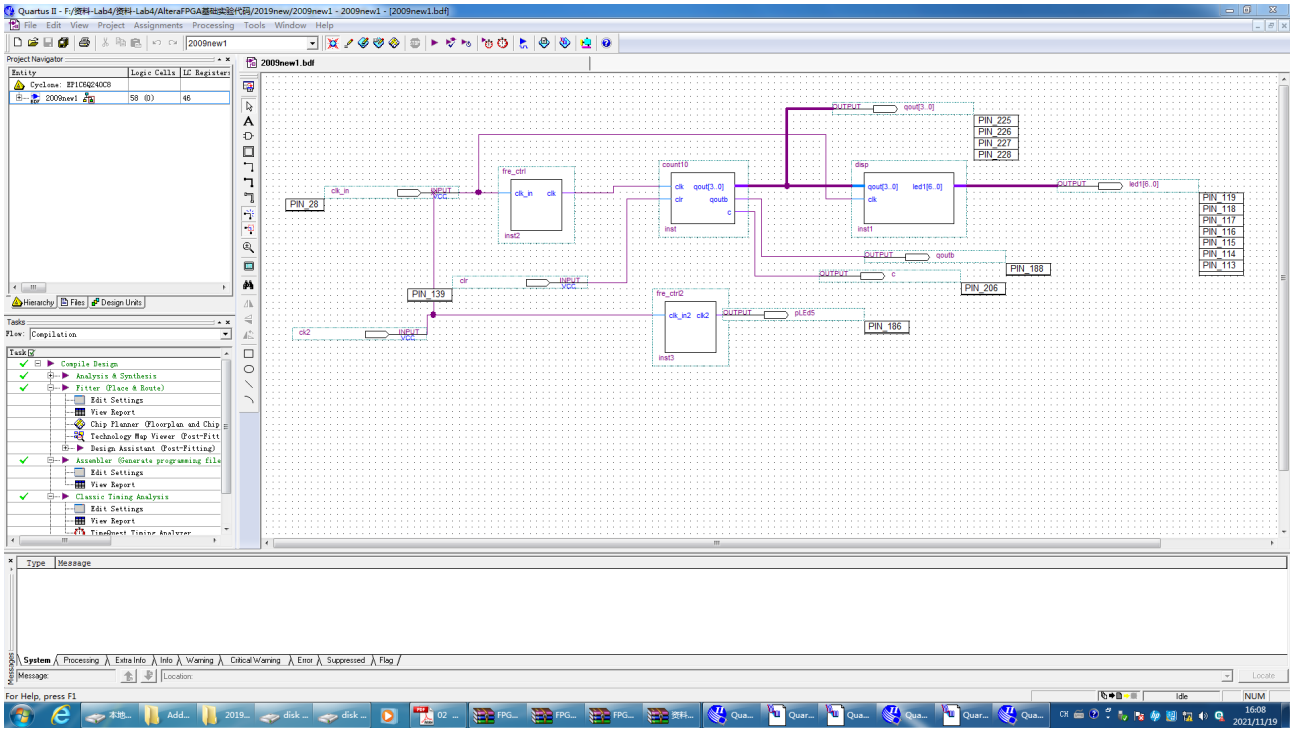

图 20: 测试设计

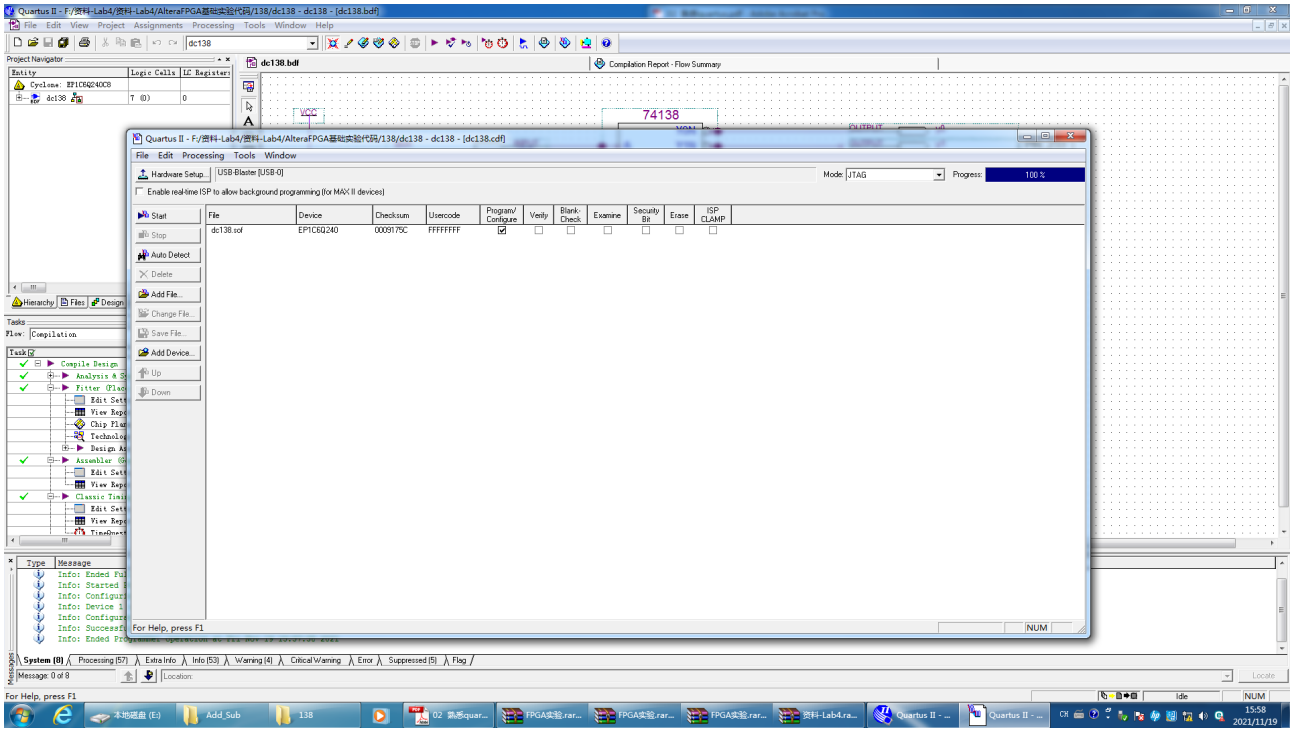

图 21: 烧录程序

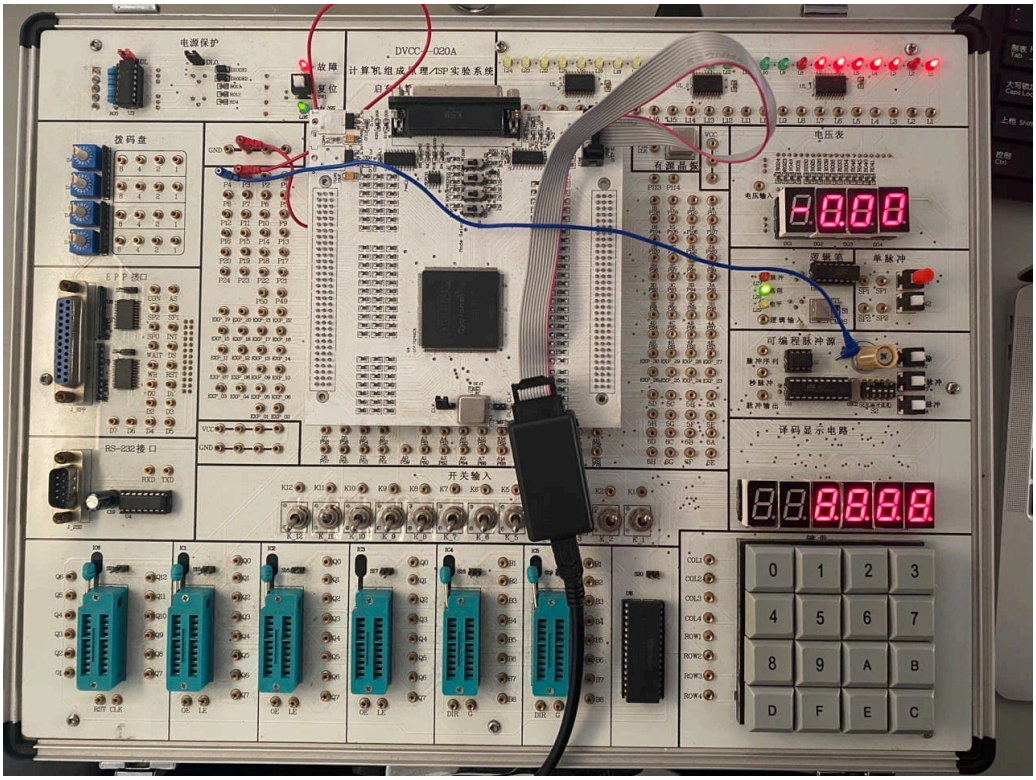

图 22: Altera 实现结果

三. 选作探索与应用设计

Vivado 实现上述 Altera 设计，由于手边没有合适的开发板，我只能在理论上分析这个结果.

每一次 run 完 Implementation 后选择 Generate Bitstream 后可以选择烧录至开发板中，后面的过程应该是较为类似的，只是 Altera 的实现更多通过可视性的操作（和电路仿真软件 Multisim 较为接近），Vivado 可以将实现交给电脑来做. 个人认为你 Vivado 在大工程项目的逻辑上更好掌握，即可以有一个明确的结构和互相调用的关系.

四. 实验总结

这次实验内容较多，包括了 Vivado 和 Altera，其中 Vivado 基本都是课后完成的，实验室内完成了 Altera 实验. 具体分析在各个部分都有，总体的感受是有些复杂的操作，我大为震撼，不过由于现在学习科研压力较大，没有能够更加深入其中做更多拓展的工作. 不过目前对整个流程有一个大致的了解还是很好的锻炼，也为以后偏硬件的科研工作做好了铺垫.

实验器材 1

- 清华科教仪器厂 TPC-ZK-II_PCI 微机接口实验装置
- EEEC-020A 计算机组成实验箱一台（自带电压表）
- RIGAL DG1022U 25MHz 双路波形发生器

- RIGOL DS1072E-EDU 70MHz 双踪数字记忆示波器
- Xilinx Vivado *HLx* 2017.4
- Altera Quartus II 9.0
- Ubuntu 20 / Windows 11 / Windows 7（Vivado 实验于 Ubuntu 平台完成，Altera 实验于 Windows 7 上完成）

参考文献

[1]　计算机硬件实验室.《数字逻辑与计算机体系结构》实验指导书[A]. 南京: 东南大学, 2021.

附录 A：实验报告 LaTeX 模板

实验报告使用自己编写的 LaTeX 模板（`SEU-Digital-Report.cls`），在基本适配 Microsoft Word 版报告的格式要求之外，增加了更多的功能，使得报告看起来更加优雅多彩.

后续升级后，报告模板将于 https://github.com/Teddy-van-Jerry/TVJ-Digital-Report 基于 MIT License 开源共享.

编译需要使用 `XeLaTeX + Biber`，封面页修改如下内容即可.

```
%% 使用实验报告模板类（字体大小 11pt 约为五号字）
\documentclass[11pt]{SEU-Digital-Report}

%%%%%%%%%%%%%%%%%%%% 报告基本信息 %%%%%%%%%%%%%%%%%%%%
\expno{四} % 实验序号
\expname{黑箱电路元件判别及参数测定} % 实验名称
\expauthor{赵舞穹} % 姓名
\expID{61520522} % 学号
\expmates{郑瑞琪} % 同组
\expmatesID{61520523} % 学号（同组）
\expmajor{工科试验班} % 专业
\explab{计算机硬件技术} % 实验室
\expdate{2021年11月19日} % 实验日期
\expreportdate{\today} % 实验日期
\expgrade{} % 成绩评定
\exptutor{冯熳} % 评阅教师
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

附录 B：Vivado 程序真伪判别

1. Vivado 程序我在 Ubuntu 20.4 LTS 平台完成，标题栏为经典的 GNOME 桌面风格，与 Windows 有很大区别.
2. 标题栏现实程序所在文件夹为 `/home/tvj/Documents/Verilog/SEU_Digital_Experiment`，这是我的 GitHub 私有项目.
3. Design Run 中显示的时间也可以辅助判断.（如图 4）

附录 C：Altera 程序真伪判别

1. Altera 程序我在实验室电脑上完成，是 Windows 7 版本，并且据我观察在课上完成 Altera 工作的不超过三组.
2. 截图右下角时间为 11 月 19 日，是我们实验的时间.
3. 我们有自己尝试的分配引脚的步骤，这其实超过了实验本身的要求.