|  |  |  |
| --- | --- | --- |
|  | **计划内容** | **计划时间（天）** |
| 1 | APB相关文献阅读 | 1 |
| 2 | 使用服务器上的Vivado软件写代码并仿真 | 3 |
| 3 | Verilog代码规范 | 1 |
| 4 | 使用Vivado进行综合 | 2 |
| 5 | 使用Vivado进行实现 | 1 |
| 6 | 使用Vivado进行后仿真 | 0.5 |

# APB相关文献阅读（1天）

阅读“IHI0011A\_AMBA\_SPEC”（着重阅读Chapter 5.2）和“dw\_apb\_uart\_db” （着重阅读Chapter 6）

任务：

什么是APB？

APB里的寄存器有什么作用？

往APB里写数据是写到哪里？读数据是从哪里读？

如何往APB的寄存器写数据？

什么是中断？

如何清除中断？（什么是写一清零？）

# 使用服务器上的Vivado软件写代码并仿真（3天）

任务：

注册服务器，在服务器上使用Linux系统

使用服务器上的Vivado软件写代码并仿真

使用服务器所需软件：

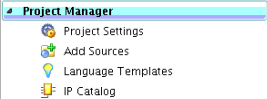
下载VNCviewer，在此软件中操作服务器。

下载FileZilla，在此软件中上传下载文件。

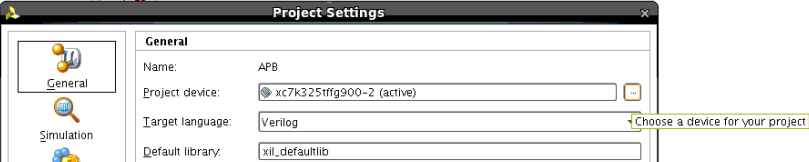
Vivado使用步骤：

新建一个文件夹，命名为APB，在文件夹里右键鼠标，选择open in terminal，输入vivado，回车，打开Vivado

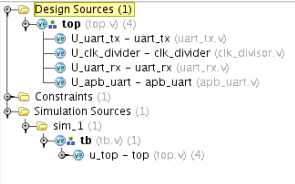
下图就是Vivado写代码的菜单，和Modelsim大同小异，下面讲几个不一样的地方



1. 在Project Settings里选择你所使用的芯片，如下图：



1. Add Sources就可以添加.v文件，添加之后需要右键top文件，选择“Set as top”将其变为top文件，右键tb文件，选择“move to simulation sources”。之后菜单变成了非常清晰有层次的下图：



1. Language Templates是一个Verilog语言指南，会告诉你规范的寄存器如何写、综合的约束如何写，可以把它当作一个Verilog字典来用。

# Verilog代码规范（1天）

阅读ppt“Verilog规范-asic“，参考Language Templates。

在下一步综合时，综合器对Verilog代码的要求更为严格，所以必须要将代码规范，否则会出现“能过仿真，却会在综合时报错”的问题。

# 使用Vivado进行综合（2天）

任务：

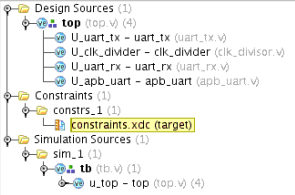
什么是综合？

如何写约束文件？

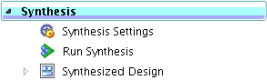
什么是网表？

步骤：

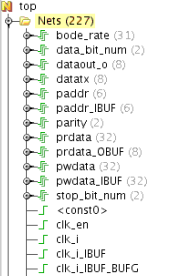
1. 先完成前仿真，确保功能正确。
2. 写约束文件，在Project Manager里选择Add sources可以添加。至于怎么写约束文件，可以在Language Templates里找找示例。成功后如下图：



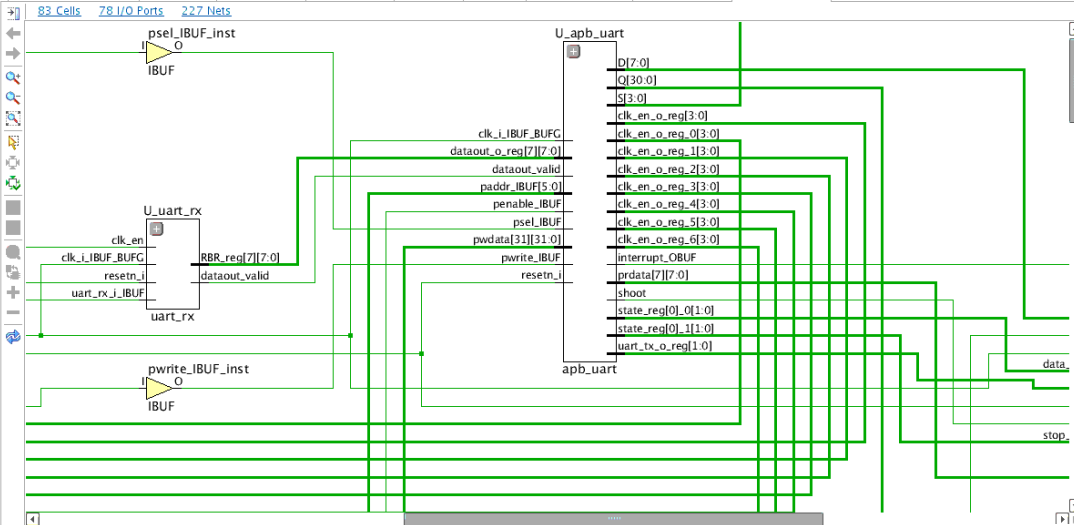
1. 综合的菜单如下图。在Synthesis Settings里选择约束文件。然后点击Run Synthesis开始综合。



1. 修改代码直到没有error和critical warning，综合才能通过。
2. 得到网表，如下图：



右键网表(Nets)，选择Schematic,可以直观看到生成的电路图，如下图：



# 使用Vivado进行实现（1天）

任务：

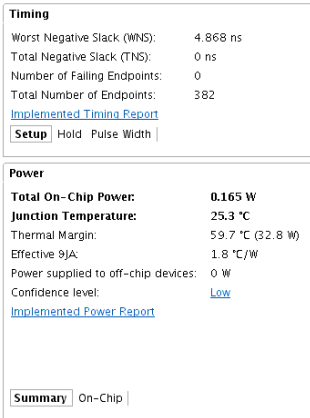
什么是实现？

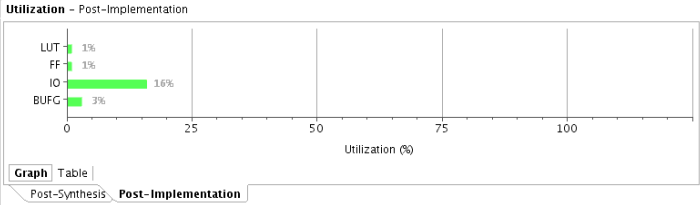
生成时间、面积、功耗报告（重点是时间报告，什么是slack，为负是什么意思）

步骤：

1. “Run Implementation”

2. 得到时间、功耗、芯片利用率的报告如下图：





3. 实现后会单出一个叫Device的窗口，即实现后的电路，如下图：



芯片实际就是这个样子，可以看到左边蓝色的部分就是你实际用了的部分，如果把这个代码烧到芯片里的话，就会烧到这些地方。

# 使用Vivado进行后仿真（0.5天）

思考：

为什么后仿真里出现了很多没见过的信号，很多前仿真有的信号却消失了？

答：

综合就是把你写好的Verilog代码，翻译成了电路图，而且它不只翻译，还会顺便给你优化电路，以满足你的时间约束，所以你会看到有一些新的你根本没有写过信号线。这些都不用担心，是综合器给你生成的，不会破坏电路功能。

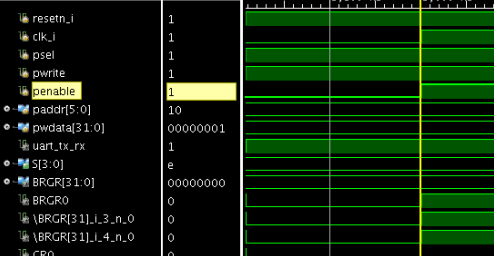
思考：

后功能仿真（Post-implementation functional simulation）与后时间仿真（Post-implementation timing simulation）的区别？

答：

区别在于是否考虑延时。

Functional，不考虑延时：



Timing，考虑延时：

