



实验预备课

2021年秋

实验预备课

- 可编程逻辑器件介绍
- 硬件编程方法与原则
- 硬件编程流程

可编程逻辑器件设计

- 可编程器件简介
- 设计原则
- 设计流程

可编程器件简介

□ 概述

- PLD是电子设计领域中最具活力和发展前途的一项技术。
- PLD能做什么呢？可以毫不夸张的讲，PLD能完成任何数字器件的功能，上至高性能CPU,下至简单的位片电路，都可以用PLD来实现。
- 目前有多家公司生产CPLD/FPGA，主要有：
ALTERA(Intel), XILINX, Lattice, Actel 。



可编程器件

□ FPGA

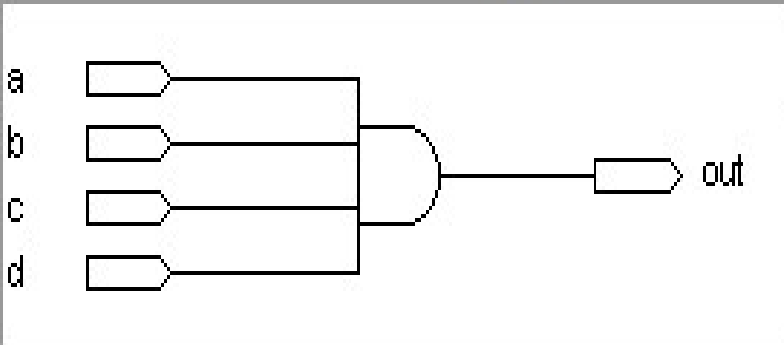
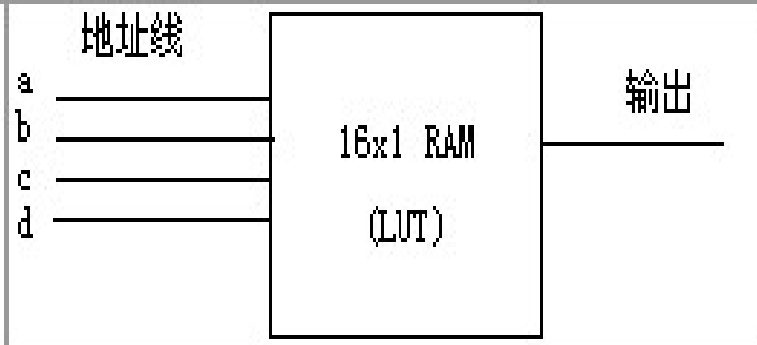
- Field Programmable Gate Array 现场可编程门阵列
- FPGA基于SRAM的架构，集成度高，以LE（包括查找表、触发器及其他）为基本单元，有内嵌Memory、DSP等，支持IO标准丰富。

查找表

□ 基于查找表（Look-Up-Table)的原理与结构:

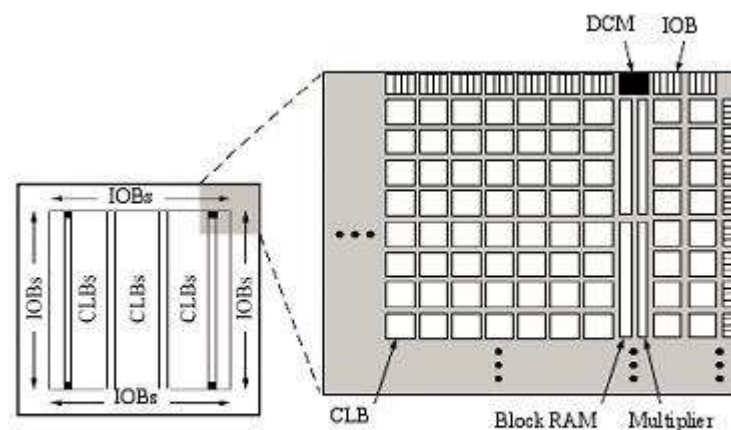
- 采用这种结构的PLD芯片如altera的ACEX,APEX系列,xilinx的Spartan,Virtex系列等。
- 查找表（Look-Up-Table)简称为LUT，LUT本质上就是一个RAM
- 工作原理
 - 当用户通过原理图或HDL语言描述逻辑电路
 - 软件会自动计算逻辑电路的所有可能的结果，并把结果事先写入RAM
 - 每输入一个信号进行逻辑运算就等于输入一个地址进行查表，找出地址对应的内容，然后输出即可。

4输入与门

实际逻辑电路		LUT的实现方式	
			
a, b, c, d 输入	逻辑输出	地址	RAM中存储的内容
0000	0	0000	0
0001	0	0001	0
....	0	...	0
1111	1	1111	1

XilinxFPGA主要部件

- 可编程输入输出单元(IOB)
- 可编程逻辑块(CLB)
- 时钟管理模块(DCM)
- 片内RAM(BRAM)
- 布线资源
- 内嵌功能单元
- 内嵌硬核



Spartan-III 系列结构

Xilinx公司产品概述

□ FPGA

- Virtex 系列
- Spartan器件系列



□ CPLD

- XC9500系列
- CoolRunner系列



□ 其他

- 配置器件SPROM (S系列 P系列)
- IP核

典型的应用领域

□ 数据采集

- 逻辑接口
- 电平接口
 - 电平不同
 - 单端到差分

□ 数字信号处理

□ IC设计验证

□ 其他类，消费，医疗，工业控制

□ 数字信号的基本上都可以用PLD实现

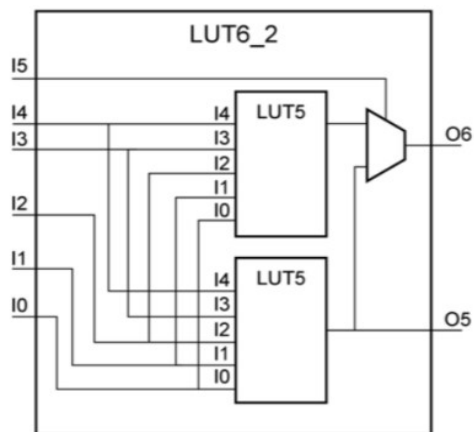


发展趋势

- 高密度，大容量，高速度
- 低成本，低电压，微功耗，微封装
- 基于IP的设计方法
 - FPGA厂家
 - 开源硬件组织
- 动态可重构
 - 通信系统
 - 重构计算机

FPGA的片内资源

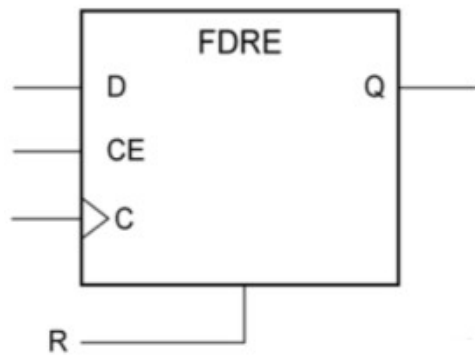
LUT结构



□ 5输入，2输出

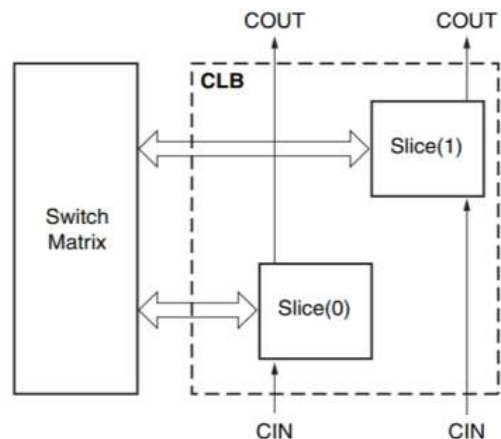
□ 6输入，1输出

FF结构（寄存器）

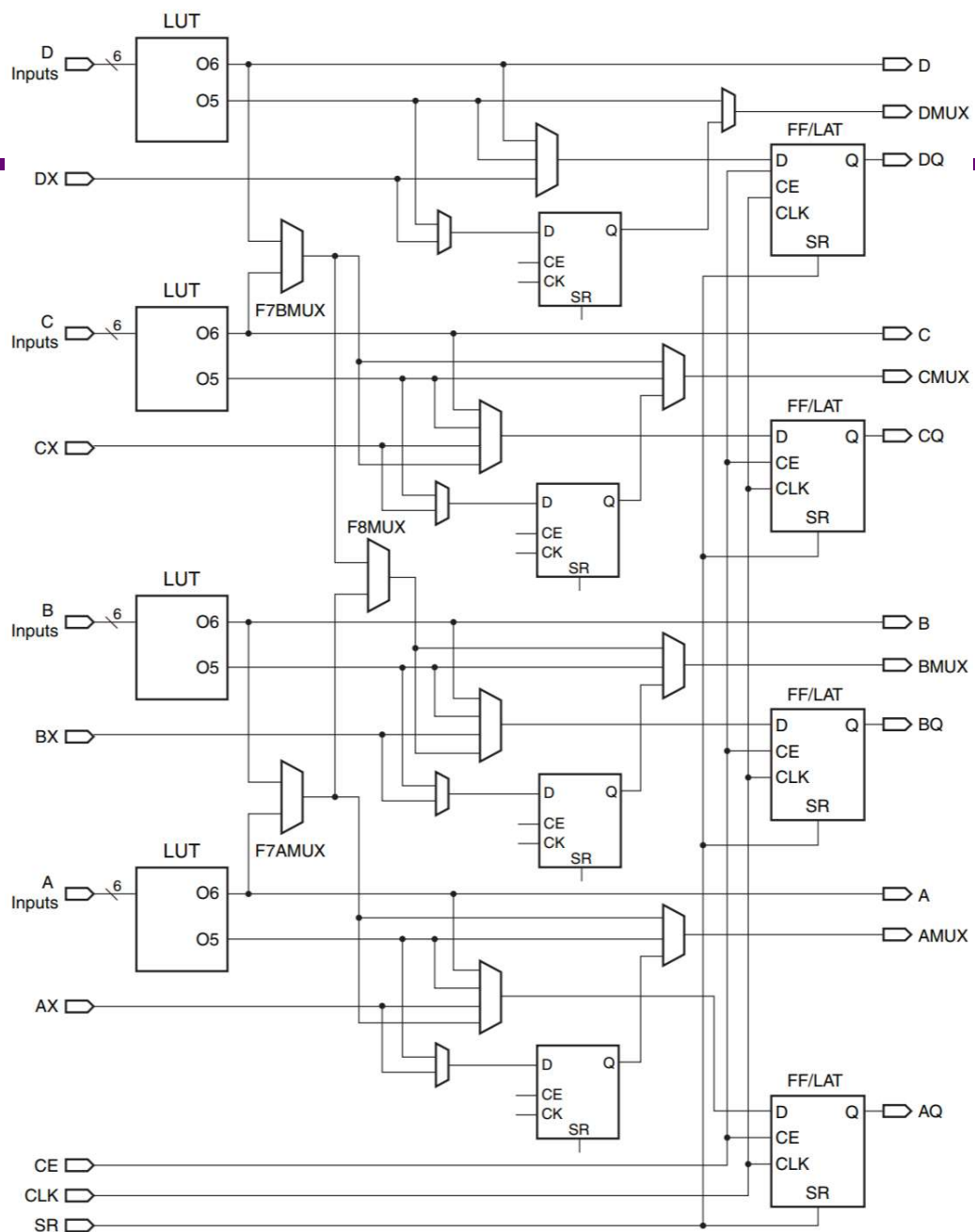


- D是数据输入
- CE是使能信号
- C是时钟信号
- Q是输出
- R是复位信号

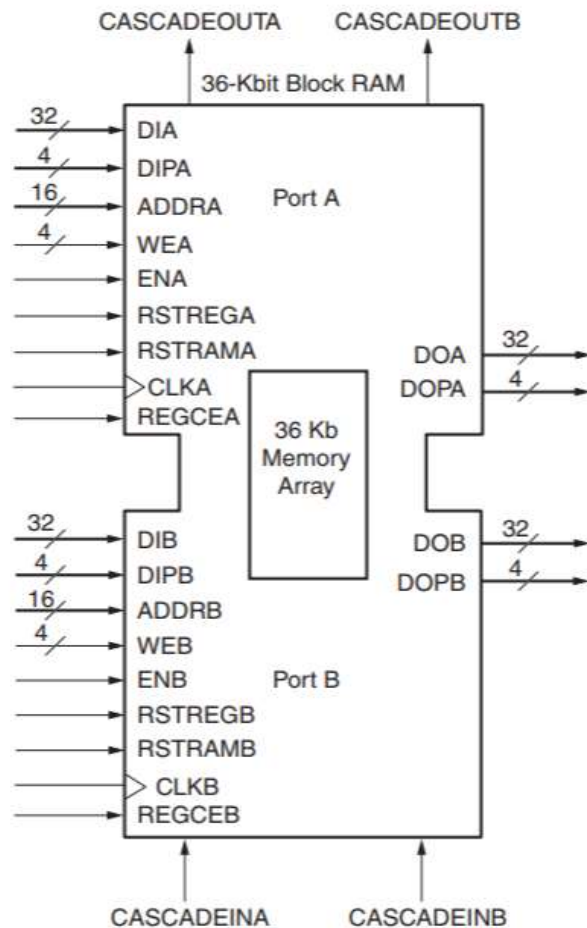
逻辑片SLICE



每两个Slice被组织成一个可配置逻辑块（CLB），最后大量的CLB之间再由开关阵列连接起来。这样的架构使得FPGA片上的LUT、FF可以自由地连接，形成一个更大规模、可配置的逻辑电路



嵌入存储器

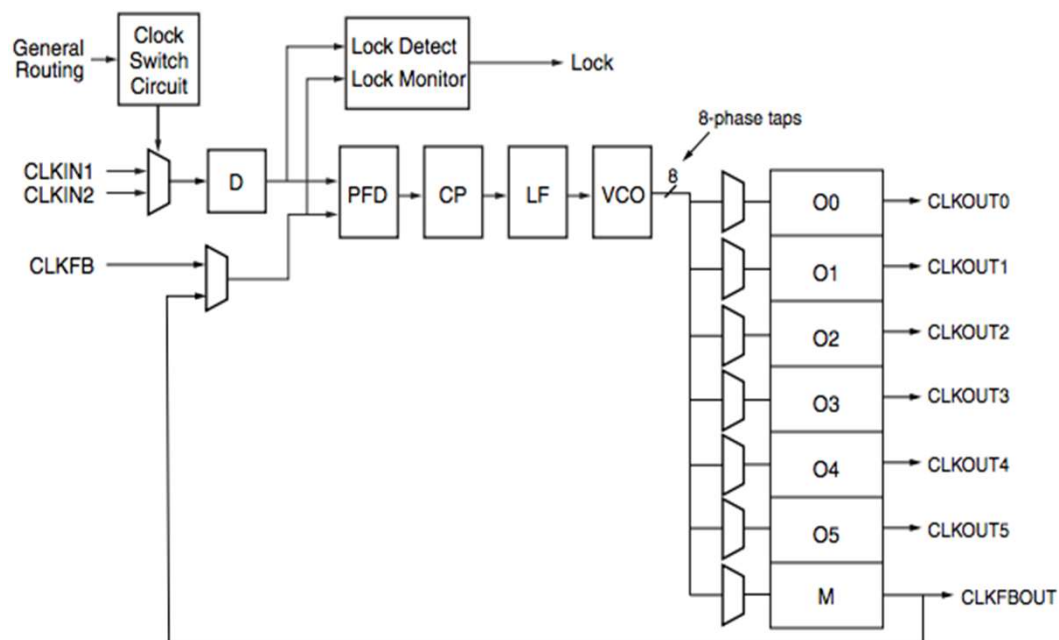


❑ Block RAM(BRAM)本质上是RAM，是FPGA内部的存储器

❑ 共有135个BRAM，总的BRAM容量为4860Kb

❑ 使用Block Memory Generator来使用BRAM

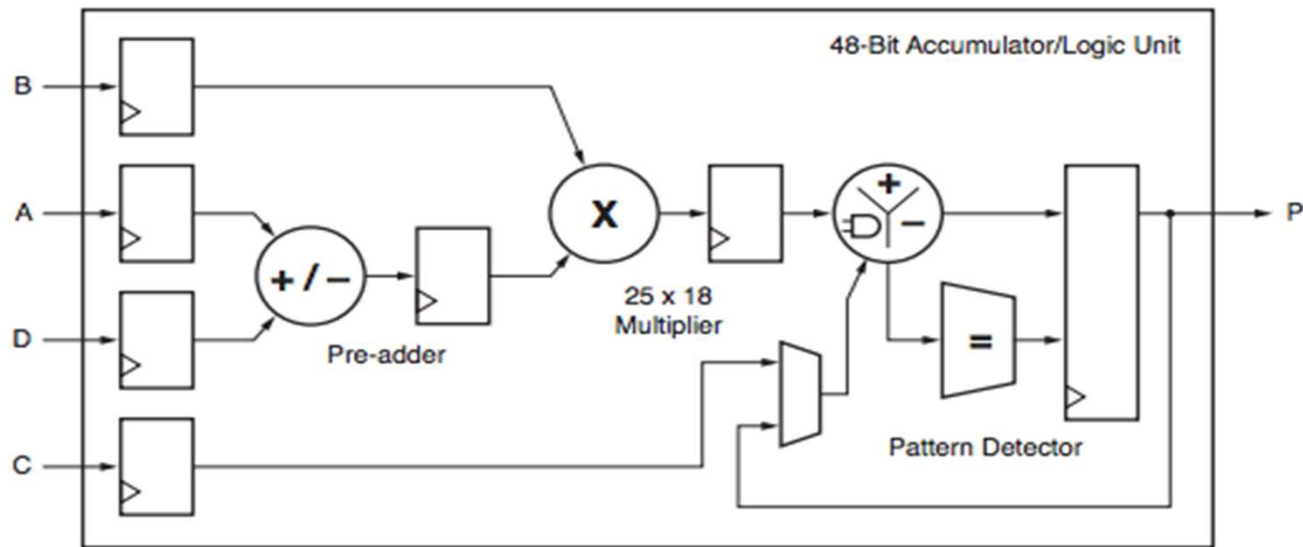
时钟管理单元



❑ 可以对输入的时钟信号进行分频、倍频，从而得到用户需要的时钟

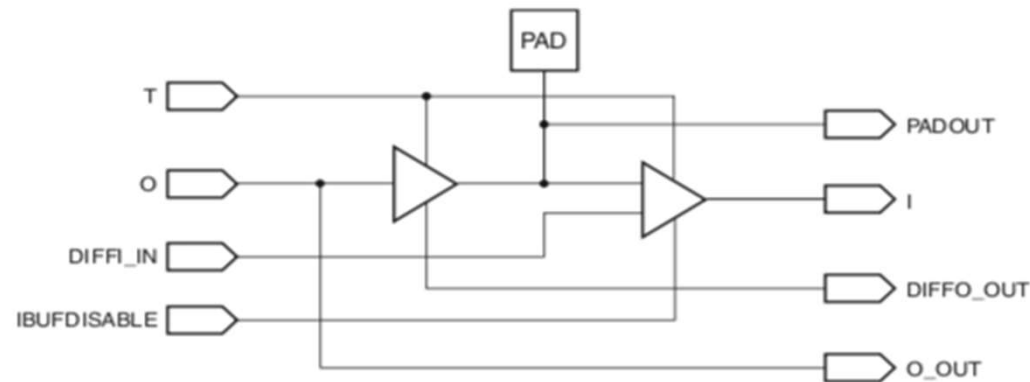
❑ 一个时钟合成器中可以设置多个分频比例，从而得到多个不同频率的时钟

数字信号处理单元



- ❑ 包含一个单周期硬件乘法器和一个加法器（也可以认为是累加器）
- ❑ 可以用来实现乘法指令操作

IO单元



- ❑ I/O口支持输入、输出信号，输出信号还受到3态门控制（可以使得引脚变为高阻态）
- ❑ 由EDA工具根据用户编写的逻辑代码中顶层信号的类型input、output和inout来自动选择

硬件设计的方法与原则

一些硬件设计原则

- 面积和速度的平衡与互换
- 功耗考虑
- 硬件原则
- 系统原则
- 同步设计原则

面积和速度的平衡与互换

- 面积和速度是数字系统设计考虑的两个重要指标，FPGA作为快速原型设计和系统验证的方法，首先就要考虑到这两个因素直接的平衡问题；
- 面积指某个FPGA设计综合之后占用的系统资源数，一般用占用的逻辑单元数量及IO接口数量来衡量，这一指标综合软件一般都能给出；
- 更小的面积通常代表更低的成本。

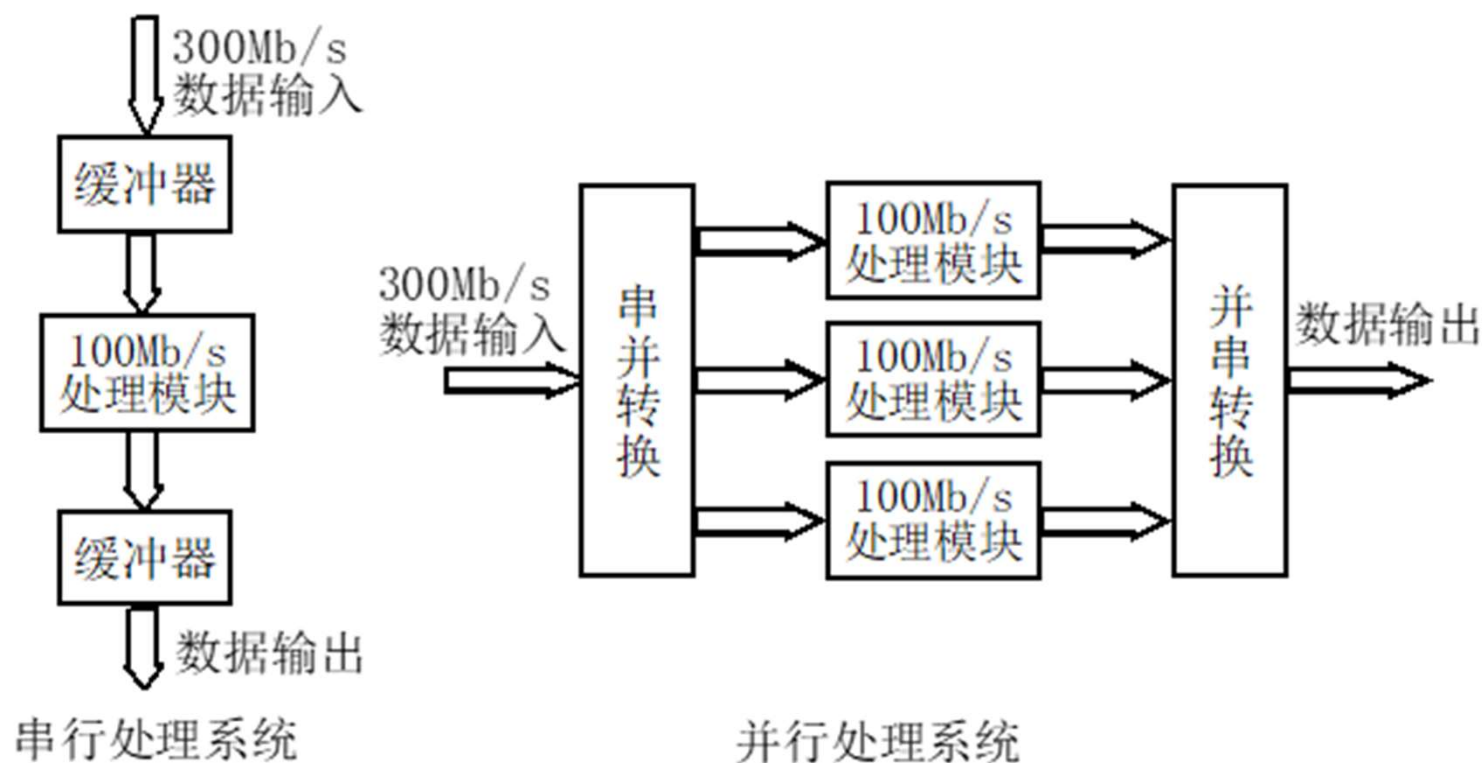
面积和速度的平衡与互换

- 简单说，速度通常指系统工作的频率，高频率常常代表高速度；
- 实际上，进行速度优化不仅仅是简单提高频率，而是要仔细考虑系统各个模块在各种工作状态下的时序要求；
- 另外可以通过并行操作提高速度；
- 在一定的工艺条件下，面积和速度常常是一对矛盾，因此需要考虑面积和速度的转换问题。

面积和速度的平衡与互换

- 将原本复用的模块进行复制，变为并行操作的模块，以牺牲面积来换取速度；
- 很多被复用的模块都是具有逻辑承接或时间先后关系的，无法直接并行化；
- 需要修改硬件设计，重新对模块做规划。

面积和速度的平衡与互换，例子



速度换面积

- 和利用面积换速度正好相反，把并行模块进行复用，以节省面积；
- 逻辑上更为简单，理论上，只要用足够的存储器，总能把并行的功能相同的模块进行复用；
- 但是要优化好存储器的管理，节省存储器，工作也并不简单。

功耗考虑

□主要考虑动态功耗：

- 动态功耗和逻辑翻转变化的频率成正比；
- 设计时要考虑尽可能不要让所有的单元同时翻转，避免引起过大的功耗；
- 例如，对于状态机的状态分配，之所以采用Gray码，另一个重要原因就是为了降低功耗；

硬件原则

- 用HDL描述硬件进行数字系统，首先应该考虑的是硬件的实现；
- 程序的可读性，必须以硬件实现为前提；
- HDL描述的是硬件的结构和各个模块之间的连接关系，在设计前应对硬件本身有清晰的了解，然后用适当的HDL进行描述；
- 注意避免软件思维

系统原则

□ 数字系统设计应该从宏观和系统全局的角度进行考虑；

- 例如对于模块等的复用和合理组织所得到的效果远比对于小部分代码的反复推敲大；

系统原则

□设计前应对所用FPGA的底层硬件资源有所了解：

- 底层可编程硬件单元
 - Xilinx的为Slice
 - Altera的为LE
- Block RAM单元
- 布线资源
- 可配置IO单元
- 时钟资源

同步设计原则

- ❑ 在目前的条件下，采用异步电路设计并不理想，而现在的FPGA芯片都是为同步电路设计优化的；
- ❑ 单纯从IC设计角度看，同步电路比异步电路更加消耗资源；

同步设计原则

- 从资源考虑，关键要优化两种资源的比例；
- 另外，同步时序电路具有没有毛刺、信号稳定等优点；
- 同步时序电路中延时的产生；
- 同步时序电路中输入的同步；

设计流程

- 设计、输入和综合
 - 原理图
 - 硬件描述语言
- 设计实现
 - 生成比特流文件
- 设计验证
 - 门级仿真
 - 下载

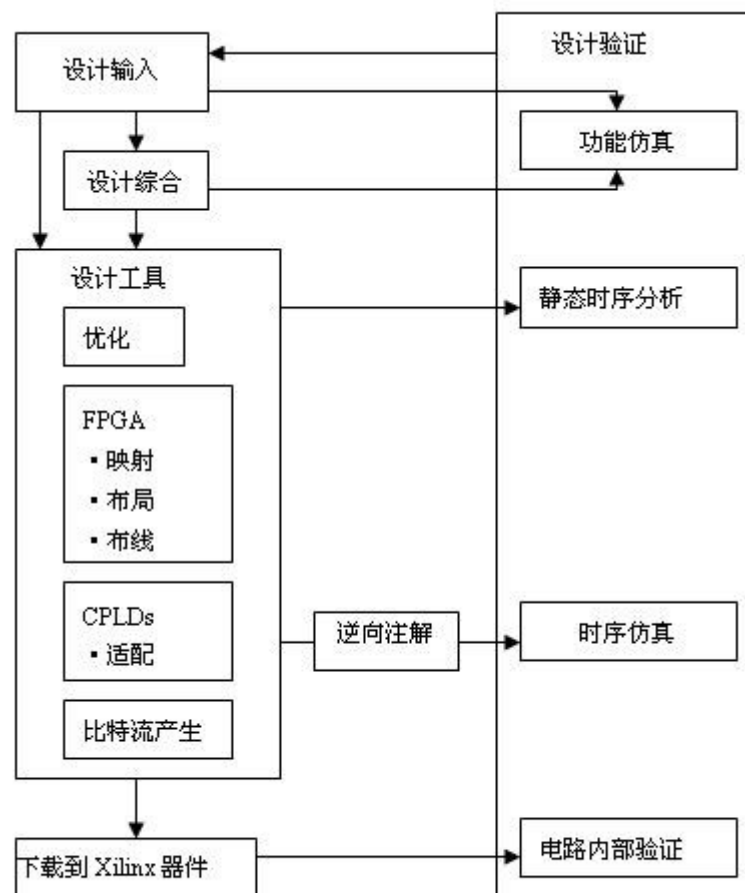


图 1.1 Xilinx 标准的设计流程

设计、输入和综合

- 方案论证、系统设计和FPGA芯片选择等准备工作
- 输入是将系统或电路以某种形式输入给EDA工具的过程
- 创建设计后可以进行功能仿真，也称为前仿真，是在编译之前对用户所设计的电路进行逻辑功能验证
- 综合就是将较高级抽象层次的描述转化成较低层次的描述

层次化设计对原理图和HDL都很重要

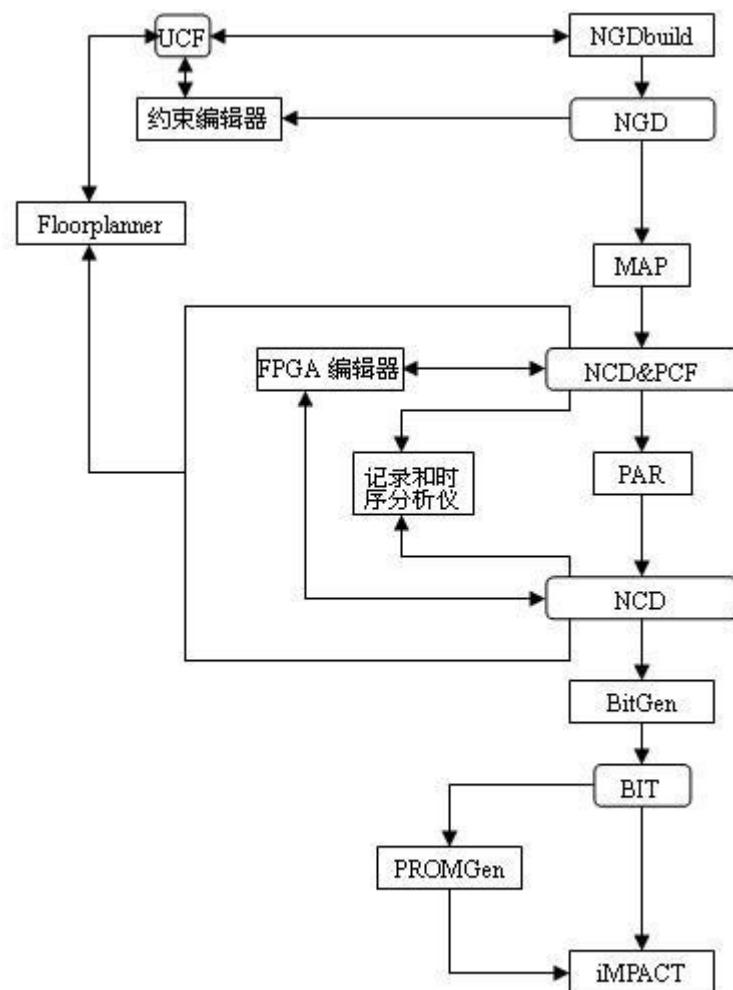
- 可以将设计概念化
- 将设计结构化
- 使调试设计更容易
- 使设计的不同部分的不同输入设计方法（原理图，HDL，本地编辑）能更容易结合
- 使更容易的更新设计，其中包括设计，实现，以及在设计过程中验证个别元件
- 减少优化时间
- 便于并行设计

约束

- 如果想要对设计中的时间参数或者布局参数进行约束，设计者可以指定映射、块布局、以及时间规范
- 映射约束
 - 指定逻辑块如何映射到可配置的逻辑块
- 模块布局
 - 块布局可限制在指定位置，可以是多个位置中的其中一个，或者是一个位置区域。
- 时序规范
 - 可以指定设计中路径的时间要求。

设计实现

□ 从逻辑设计文件映射或适配到指定的器件开始，到物理设计布线成功并生成比特流文件时结束



设计仿真

□ 综合后仿真

- 把综合生成的标准延时文件反标注到综合仿真模型中去，可估计门延时带来的影响。

□ 时序仿真与验证

- 也称后仿真，是指将布局布线的延时信息反标注到设计网表中检测有无时序违规板级仿真与验证

□ 板级仿真

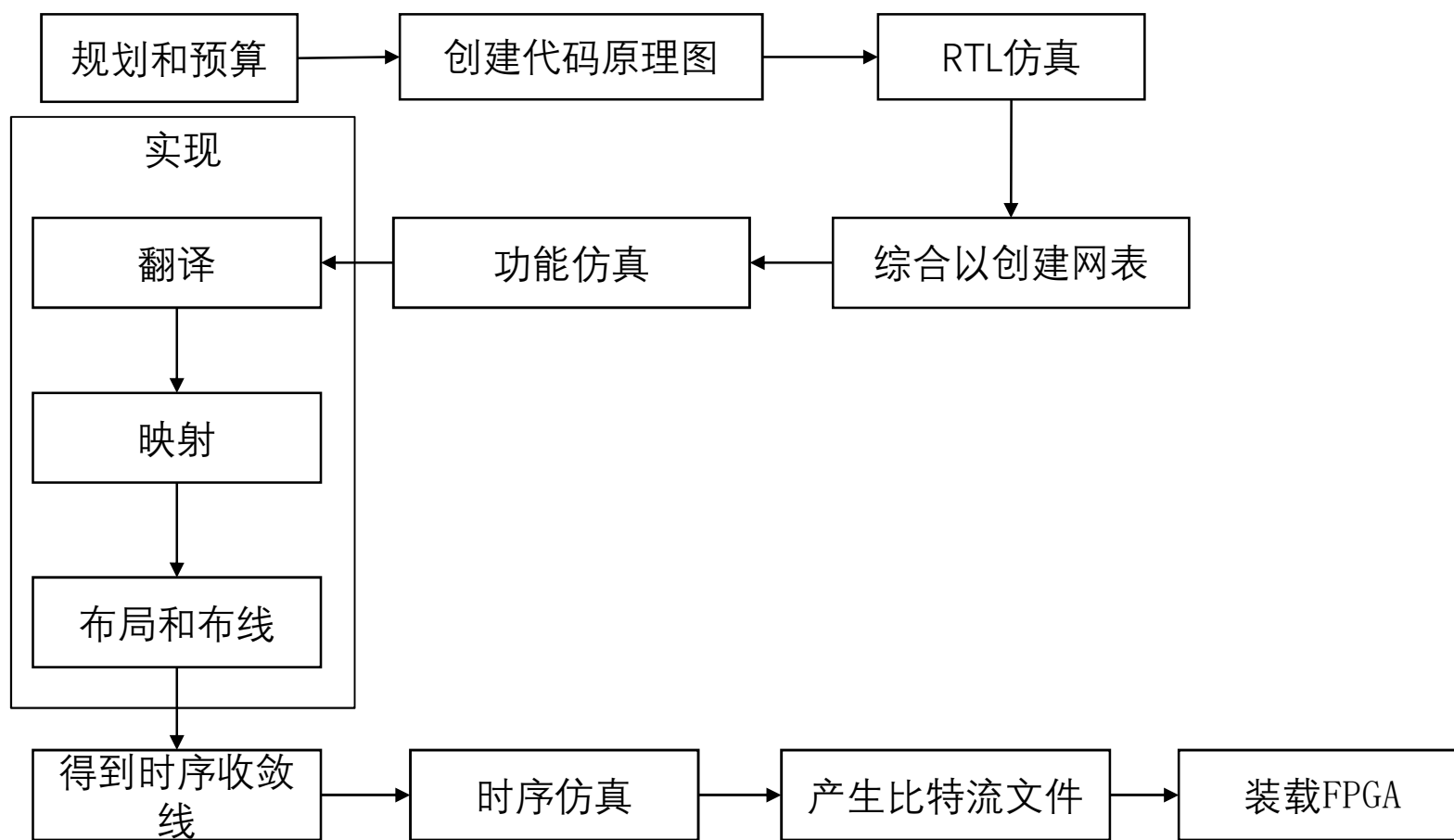
- 主要应用于高速电路设计中，对高速系统的信号完整性、电磁干扰等特征进行分析，一般都以第三方工具进行仿真和验证。

□ 芯片编程与调试

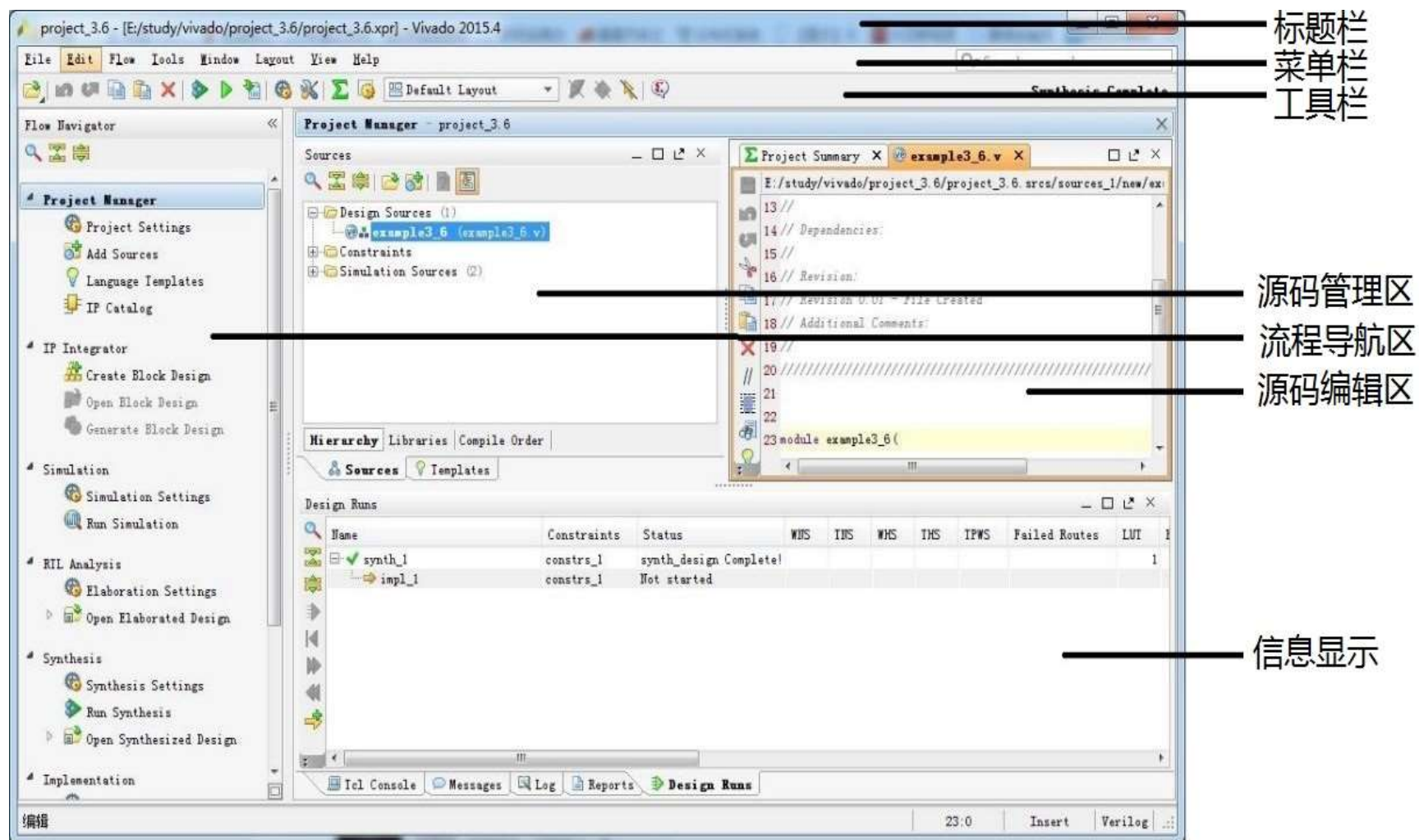
- 将编程数据下载到FPGA芯片中

Vivado进行硬件设计的 流程

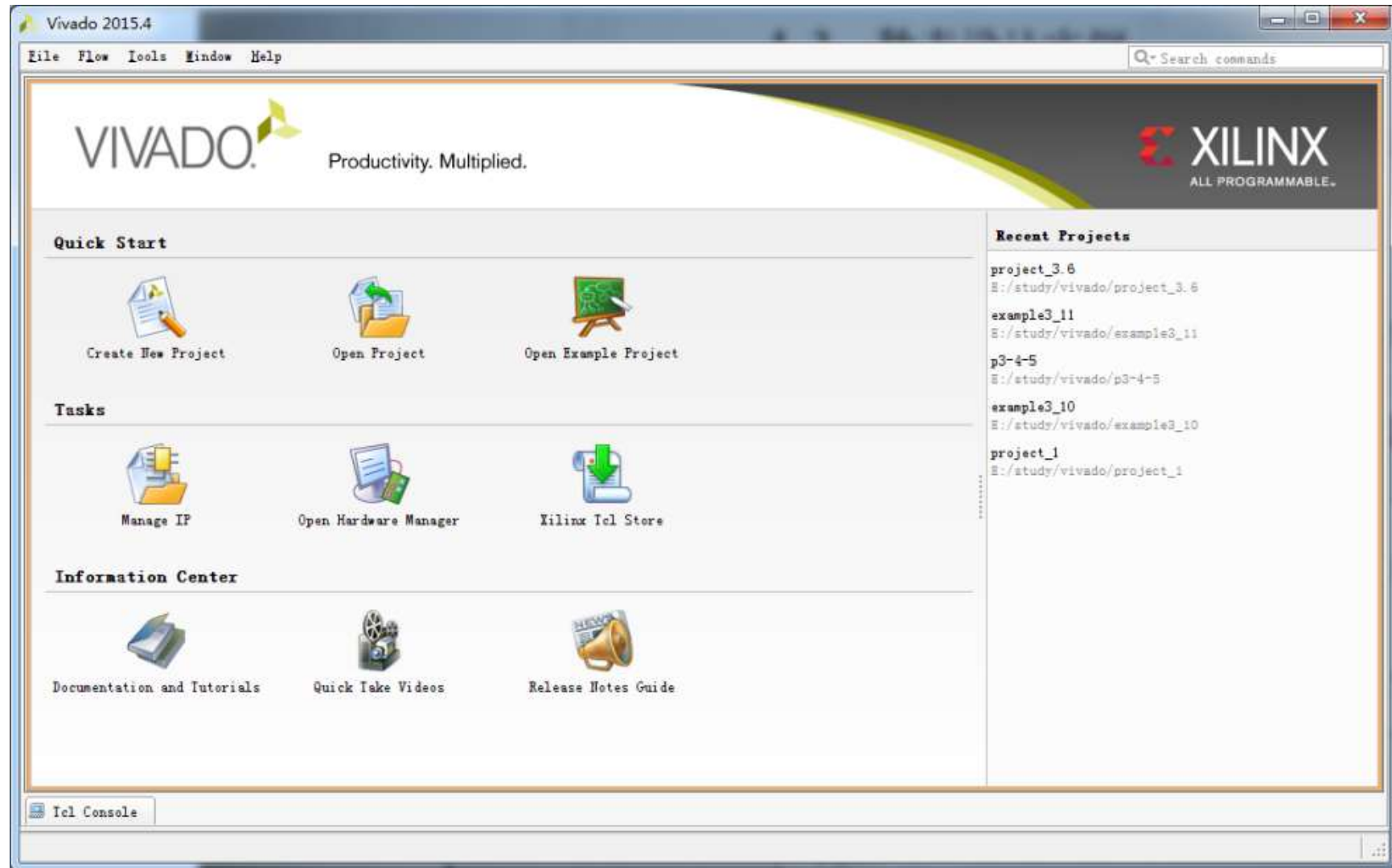
开发流程



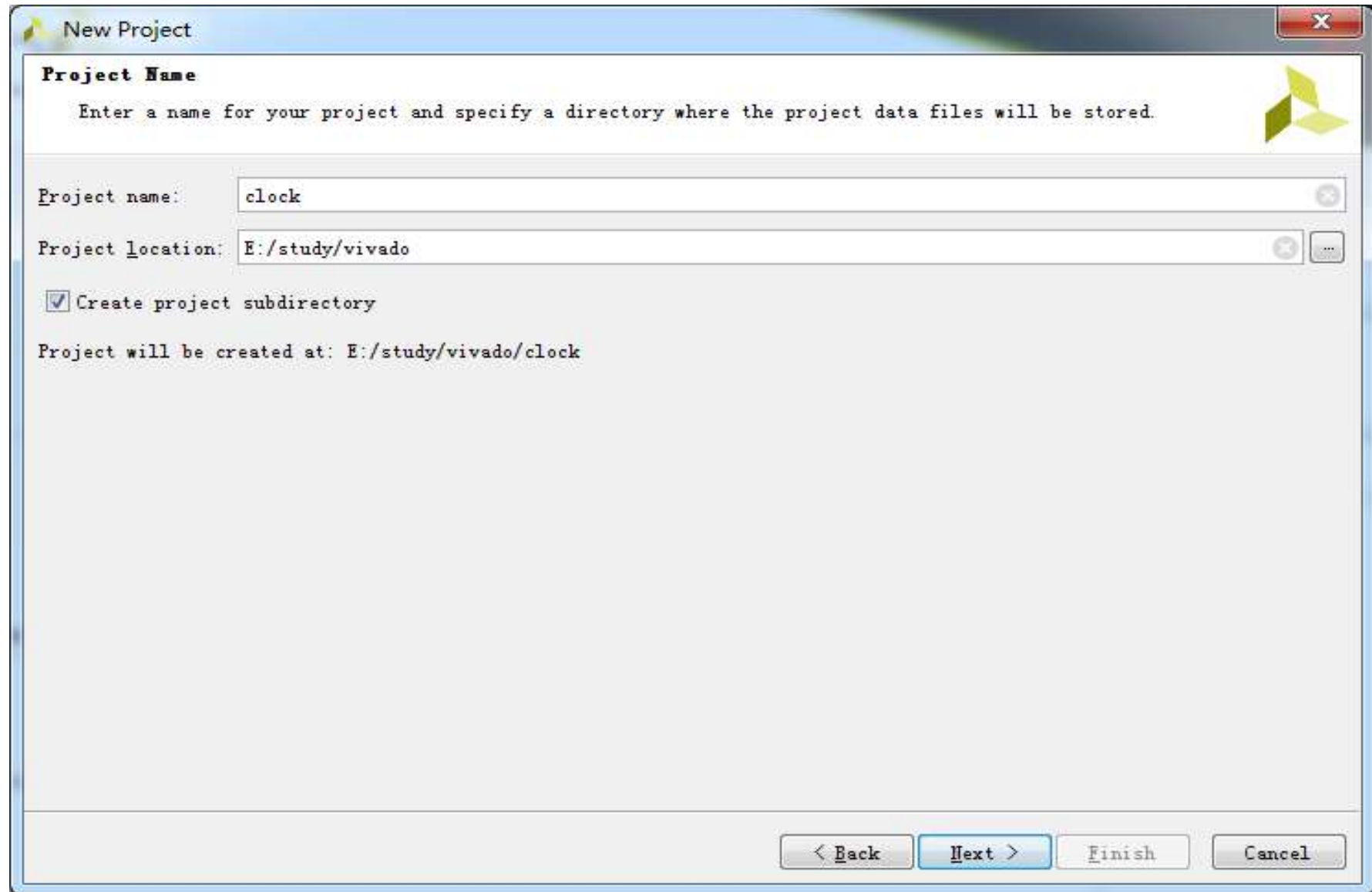
用户界面



创建空白项目



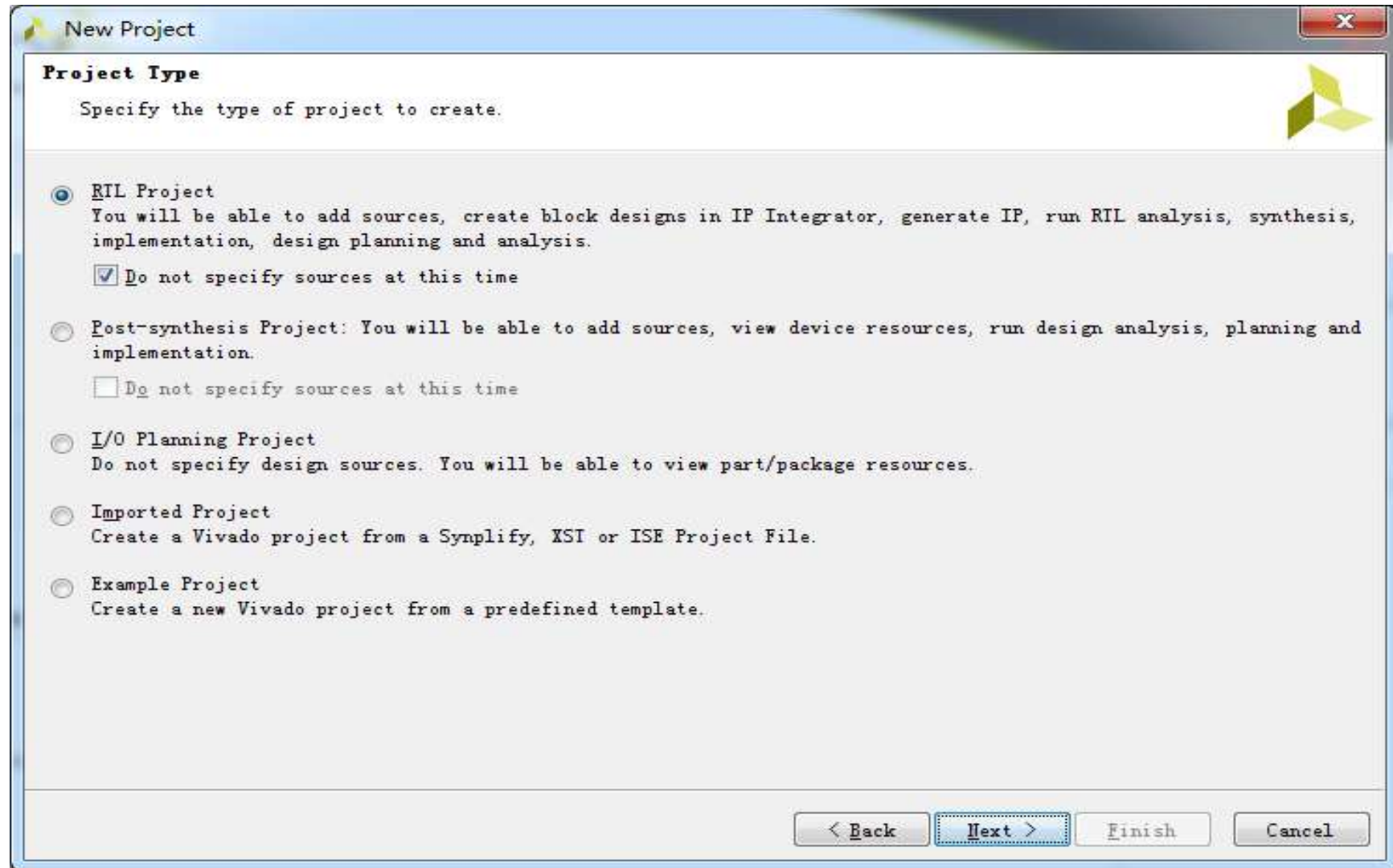
设定项目名称和位置



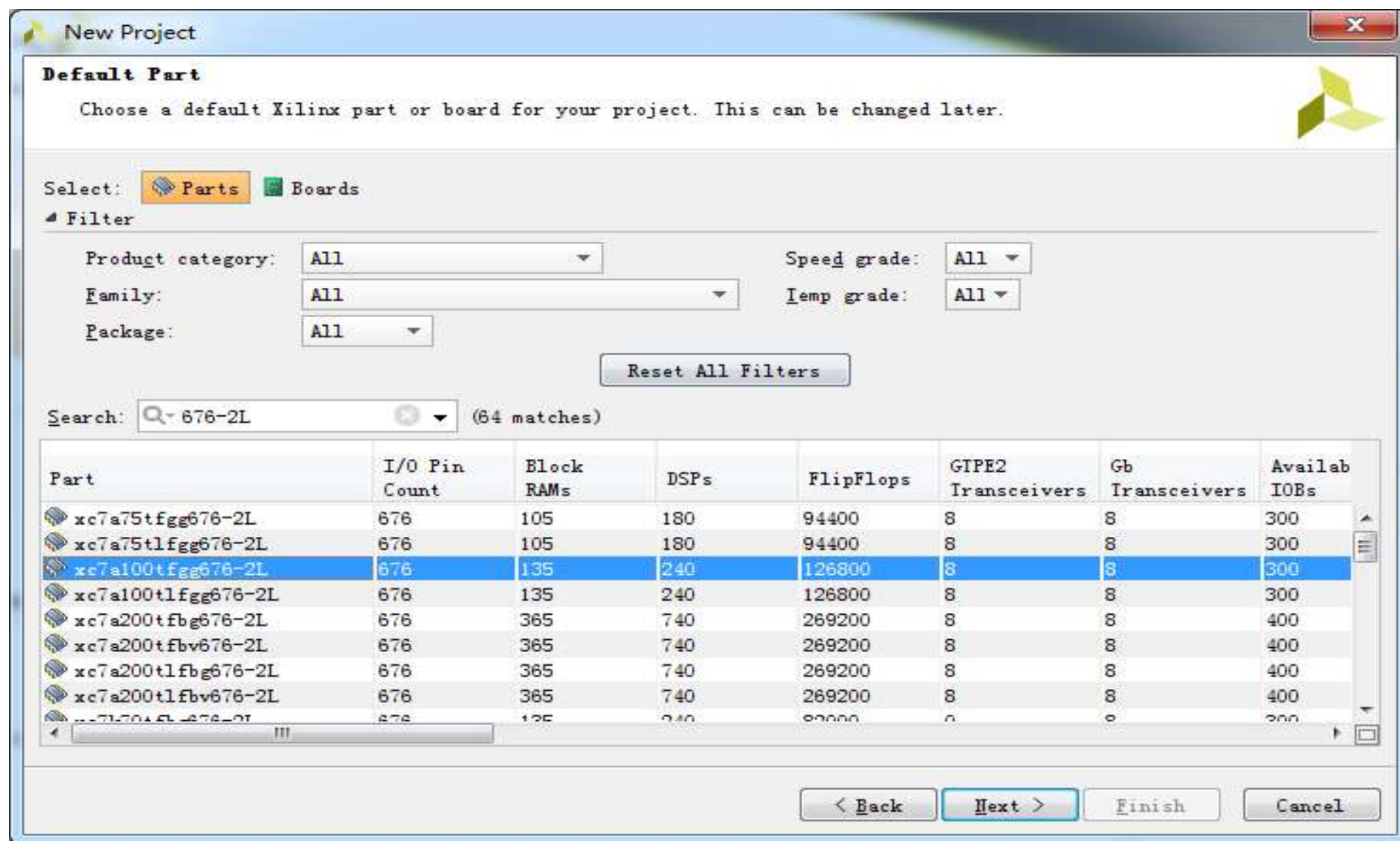
The image shows a 'New Project' dialog box from a software application. The title bar is blue and contains the text 'New Project' and a close button. The main area is white and contains the following elements:

- Project Name**: A section header with a small Vivado logo to its right.
- Instructions**: A line of text: 'Enter a name for your project and specify a directory where the project data files will be stored.'
- Project name:** A text input field containing the text 'clock'.
- Project location:** A text input field containing the text 'E:/study/vivado'. To the right of this field is a small '...' button.
- Checkbox**: A checked checkbox labeled 'Create project subdirectory'.
- Summary**: A line of text: 'Project will be created at: E:/study/vivado/clock'.
- Buttons**: At the bottom right, there are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

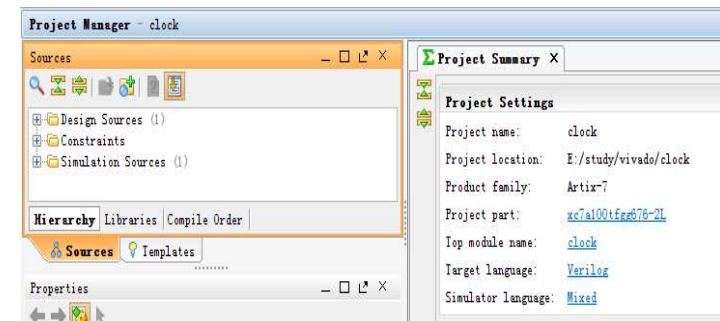
设定项目类型



项目设定



项目信息汇总与空白项目

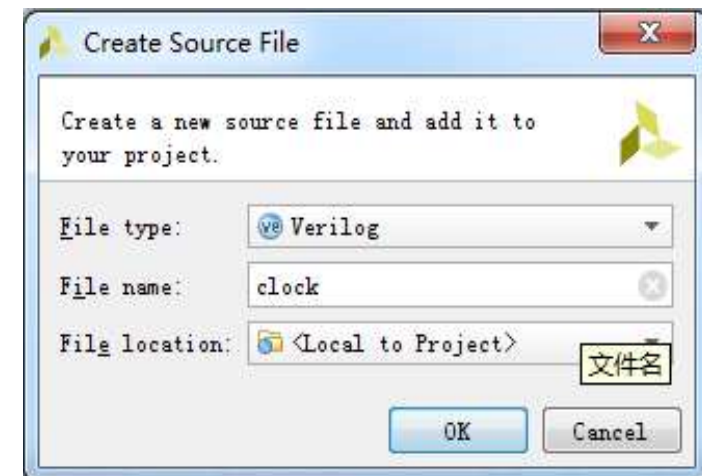
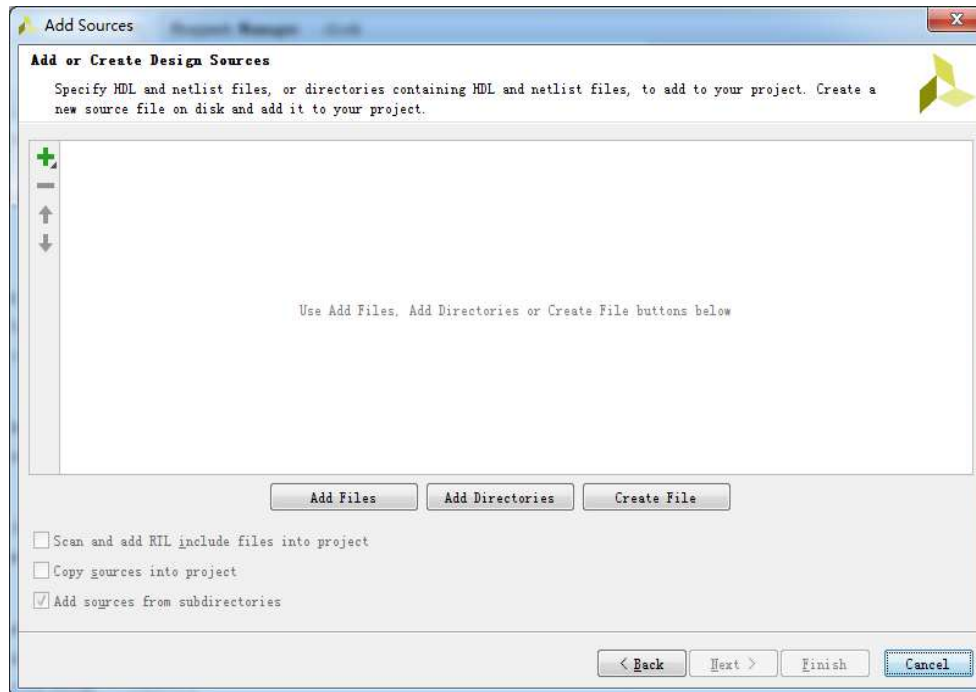


添加源文件

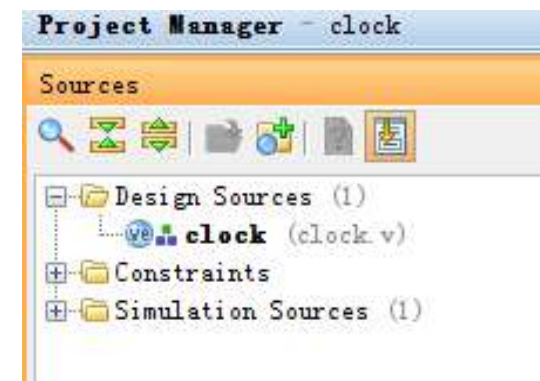
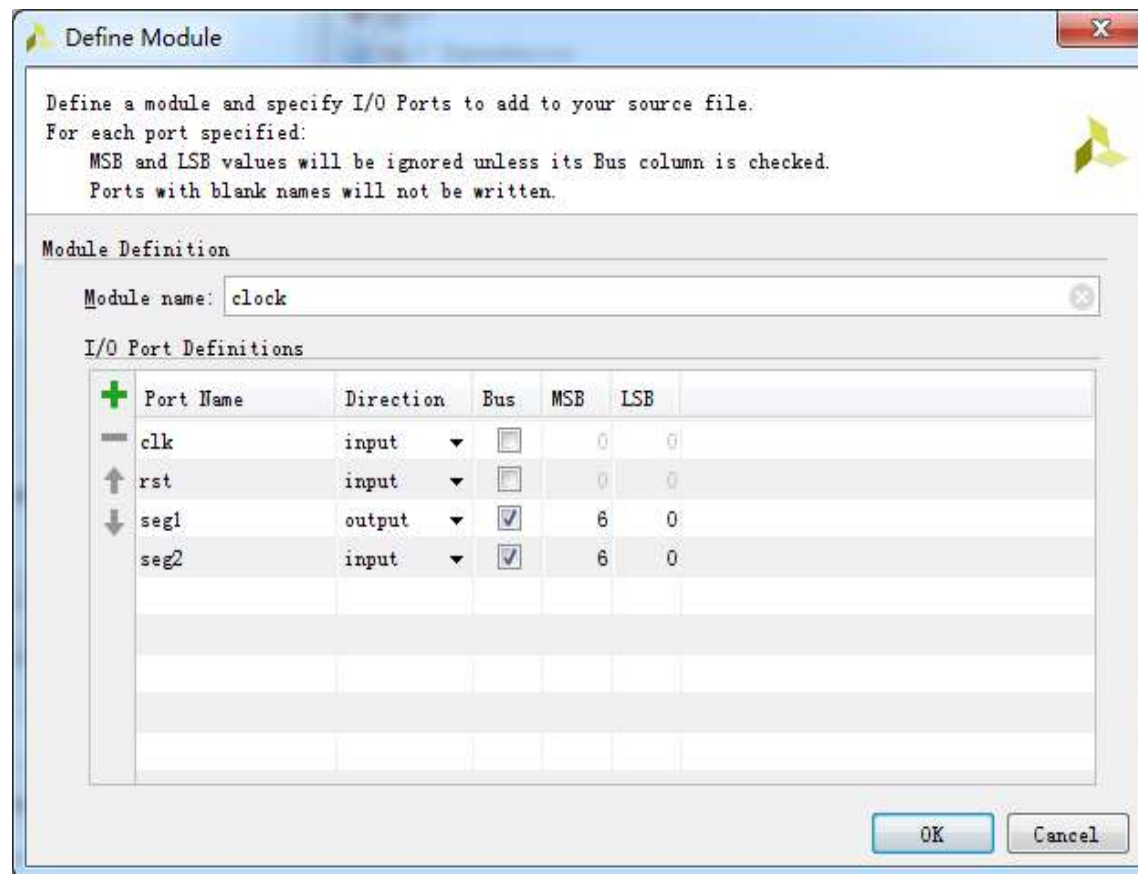
- 在左边管理区单击Add Sources，选择 New Source，出现创建源文件向导的选择源码类型对话框，在此选择 Add or Create design sources。



添加文件，创建新文件



模块定义



新文件添加完成

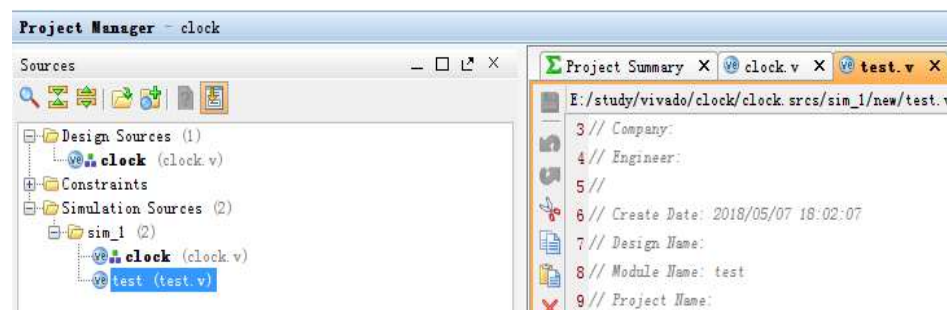
综合与仿真



综合操作

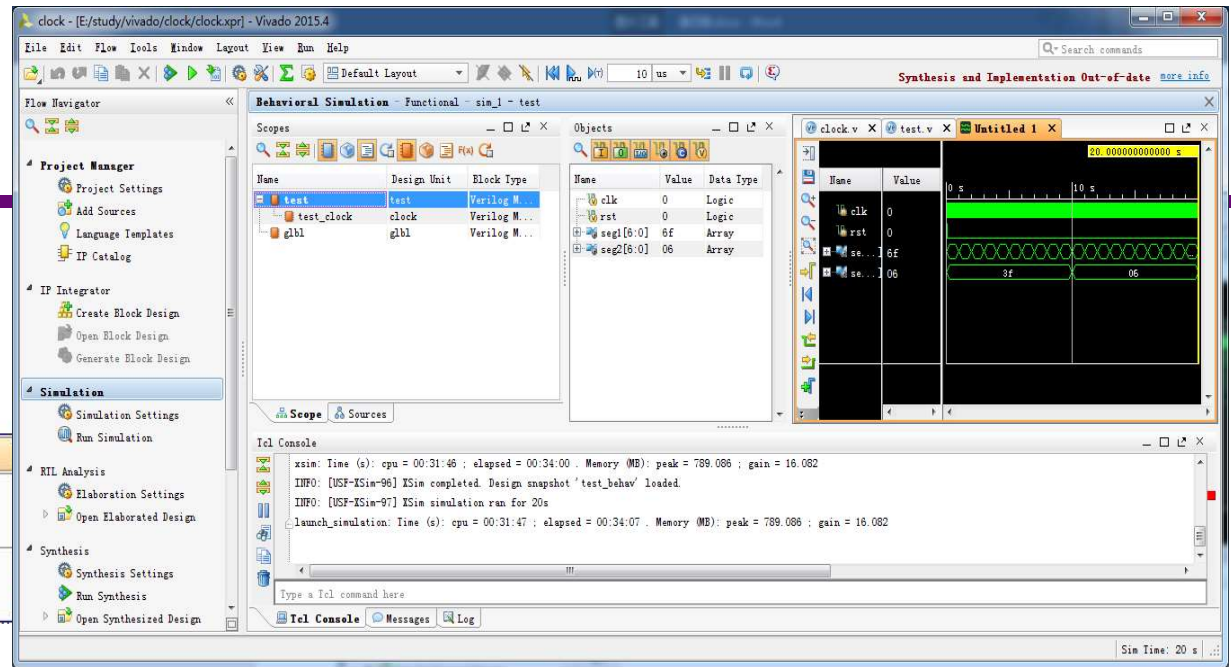
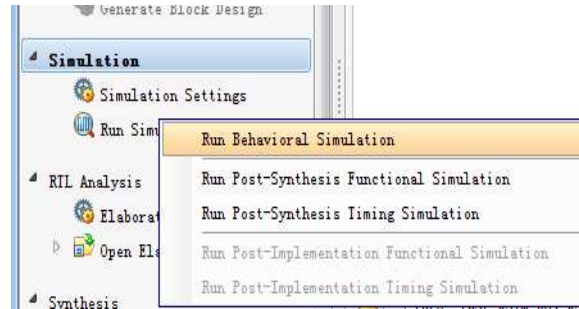


创建仿真代码



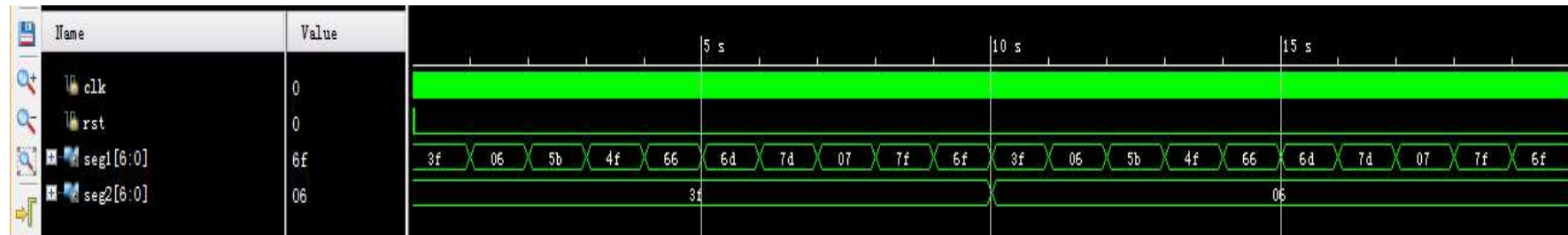
仿真源码创建完成

仿真



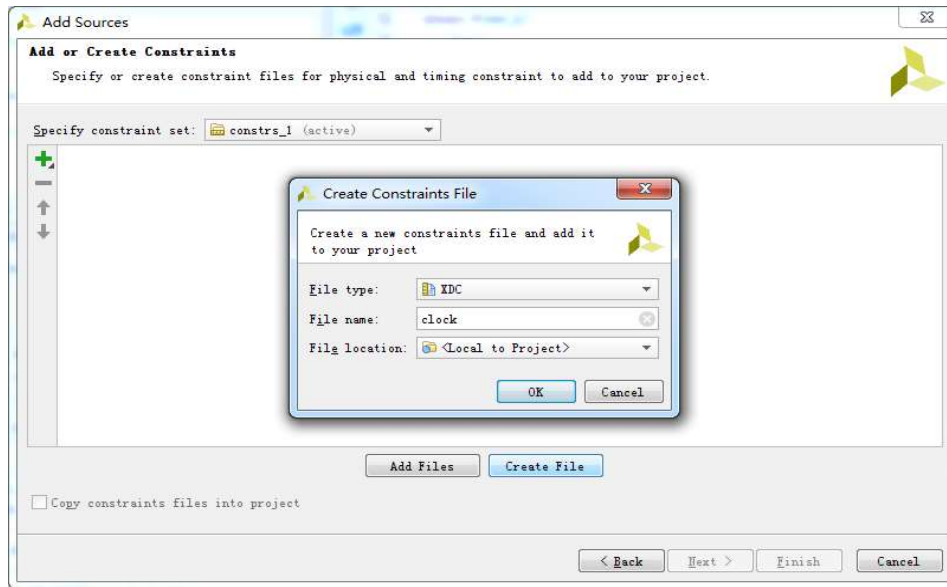
仿真开始

仿真结果窗口

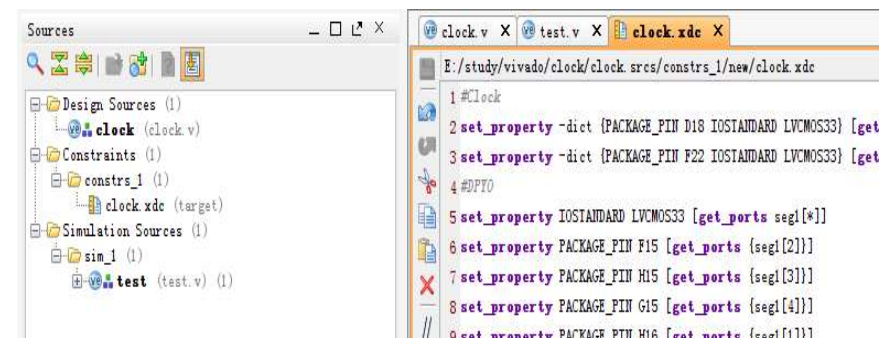


仿真结果

添加约束（样例项目已经有约束）



添加约束文件



编辑约束文件

实现，生成可配置文件



实现按钮



生成可配置文件（bit文件）

本地装载

上传设计文件

选择文件

未选择任何文件
请选择.bit设计文件

写入实验FPGA

串口参数设置

串口选择

CPLD串口控制器

⚠开发板每次重启后需要重新选择

应用

将板上的 Micro USB 接口连接到电脑，可检测到一个虚拟串口。如果在Windows系统上不能自动识别，可尝试下载[驱动文件](#)，并用更新驱动程序向导安装。

Flash与RAM读写

存储选择

Flash

容量 8MB

起始地址

0x

(Byte)

起始地址应当按16位对齐（即为偶数）

读取数据

0x

读取长度

(Byte)

读取

读取长度应当按16位对齐（即为偶数）

写入数据

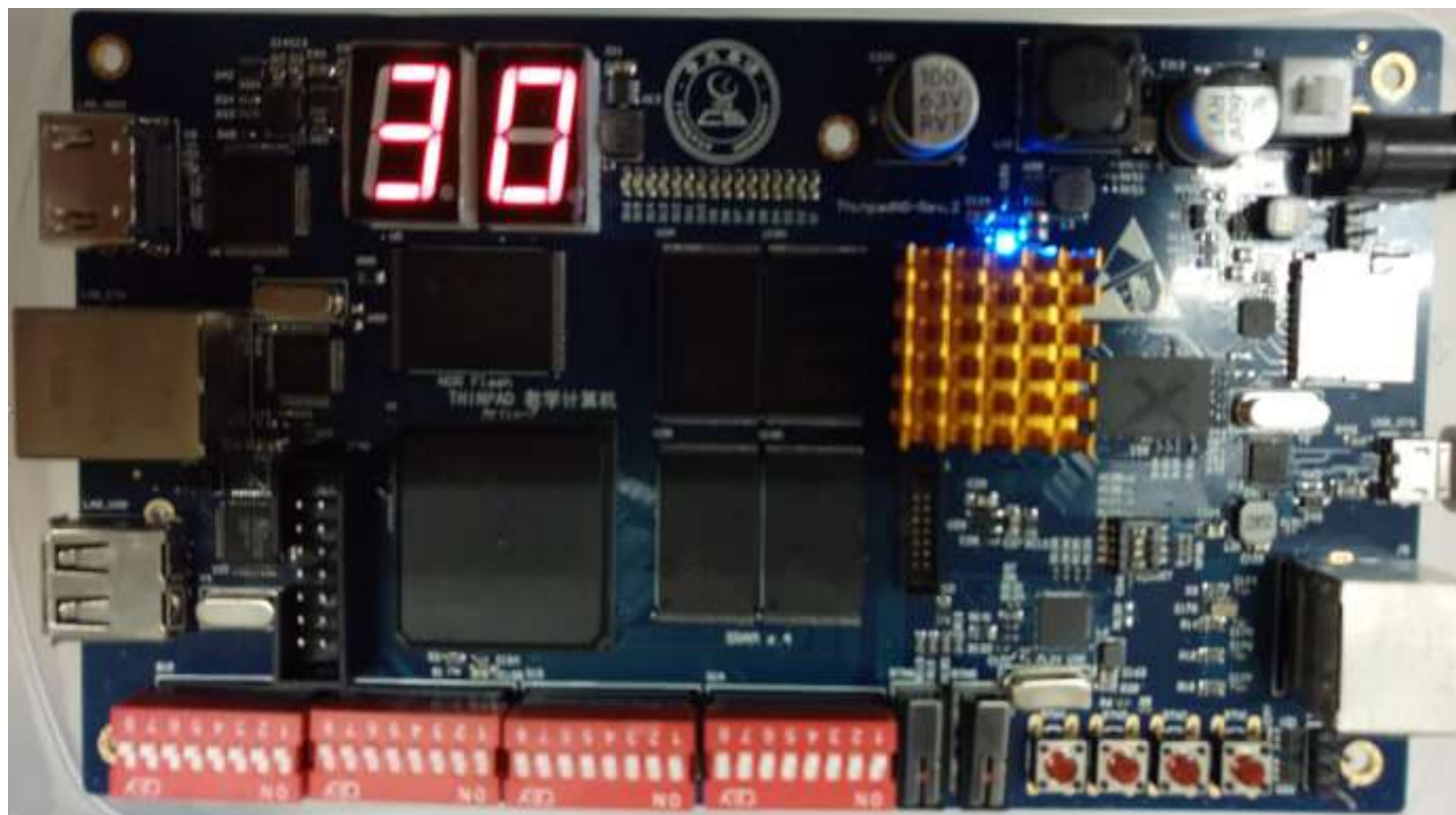
选择文件

未选择任何文件

写入

写入长度为数据文件大小，按16位对齐（即为偶数）

本地执行



远程装载

ThinpadCloud

文件上传

工作区域

admin

登出

Thinpad rev.3

ThinPAD-Cloud 教学计算机硬件平台，硬件版本3。

参考文件：
[工程模板下载（含引脚约束） rev.3](#)

开发板分配

自动分配

Bitstream

选择文件

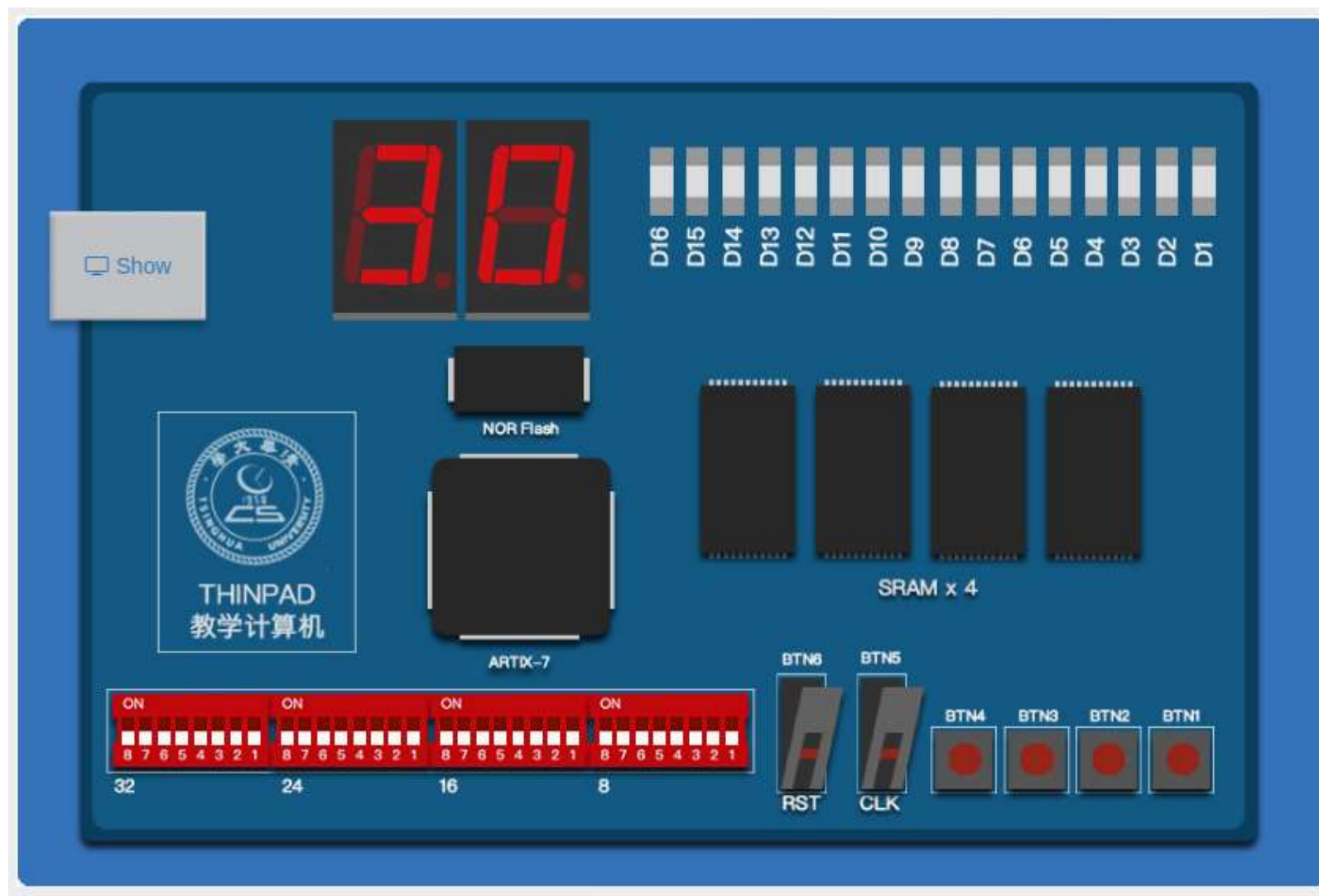
clock.bit

编译得到的 .bit 文件

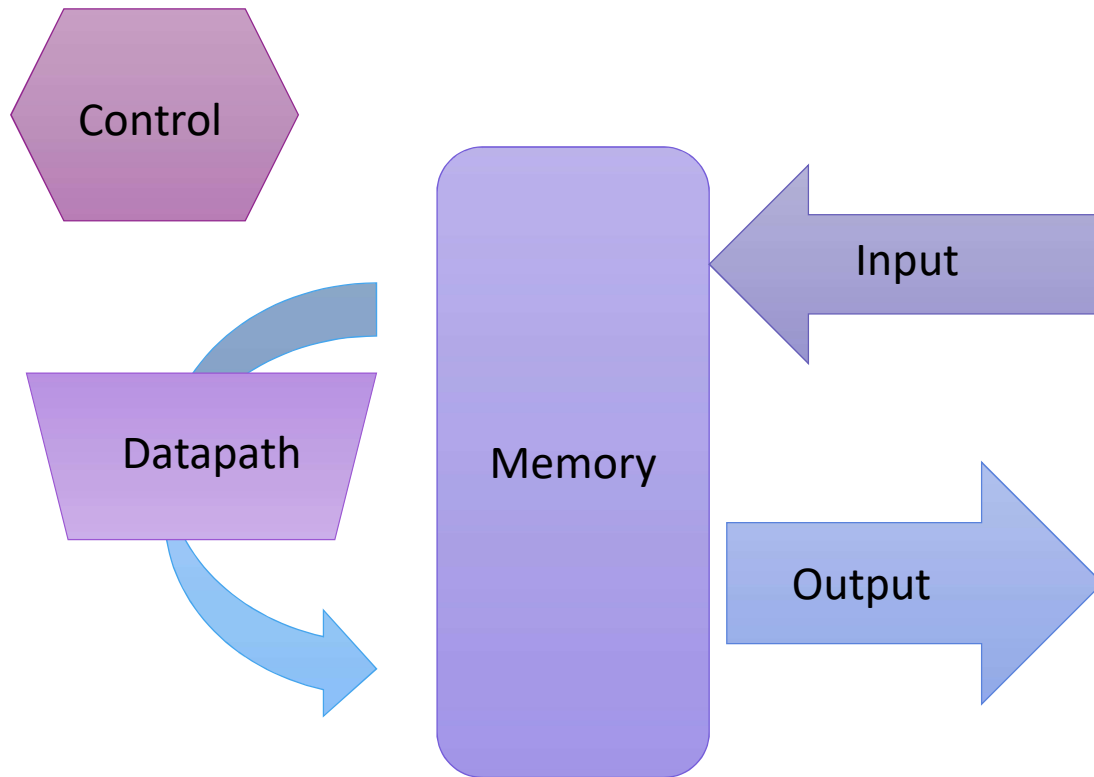
文件信息	clock;UserID=0xFFFFFFFF;Version=2018.3
编译时间	2019/06/24 16:51:45

上传并开始

远程执行

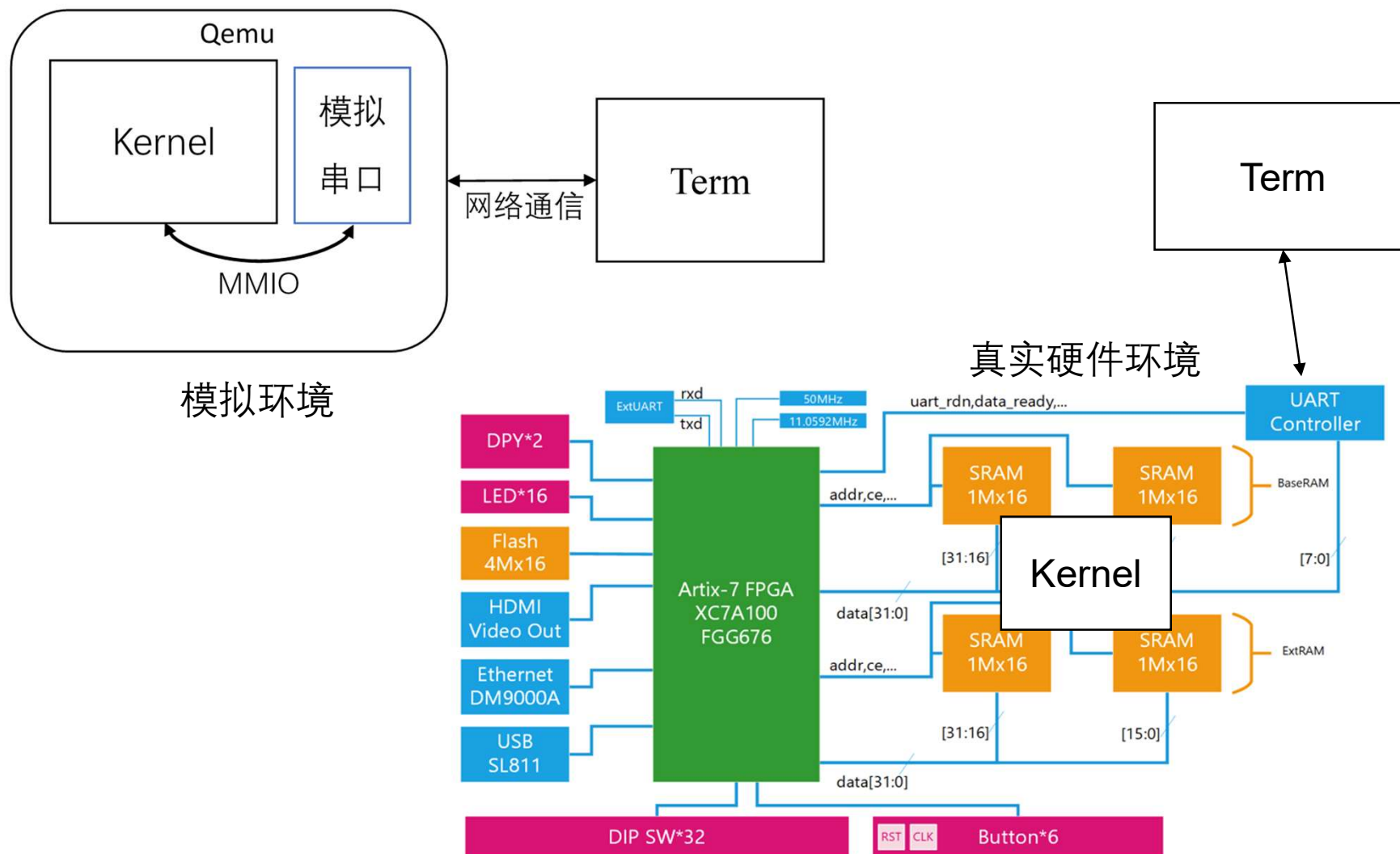


计算机运行机制



- ❑ Datapath: 完成算术和逻辑运算，通常包括其中的寄存器。
- ❑ Control: CPU的组成部分，它根据程序指令来指挥datapath, memory以及I/O运行，共同完成程序功能。
- ❑ Memory: 存放运行时程序及其所需要的数据的场所。
- ❑ Input: 信息进入计算机的设备，如键盘、鼠标等。
- ❑ Output: 将计算结果展示给用户的设备，如显示器、磁盘、打印机、喇叭等。

kernel, term, qemu的关系



监控程序的地址空间划分

虚地址区间	说明
0x80000000-0x800FFFFFFF	Kernel代码空间
0x80100000-0x803FFFFFFF	用户代码空间
0x80400000-0x807FFFFFFF	用户数据空间
0x807F0000-0x807FFFFFFF	Kernel数据空间
0x10000000-0x10000008	串口数据及状态

串口寄存器位定义

地址	位	说明
0x10000000（数据寄存器）	[7:0]	串口数据，读、写地址分别表示串口接收、发送一个字节
0x10000005（状态寄存器）	[5]	状态位，只读，为1时表示串口空闲，可发送数据
0x10000005（状态寄存器）	[0]	状态位，只读，为1时表示串口收到数据

参考网址

□ <https://www.fpga4fun.com/>

□ <https://timetoexplore.net/>

□ 上面两个网站有比较复杂的例子和详细的器件说明

□ 没有学过数字逻辑课程的同学，请务必尽快自学这部分相关的内容，注意下面的基本概念

- 注意组合逻辑的特点
- 注意时序逻辑的特点
- 时钟起到什么作用
- 什么是时延

谢谢