



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第一讲 (1.1 课程介绍)

2025年3月4日

授课团队介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS



■ 潘明忠（首席教授）

- 博士，研究员，电路与系统专业
- 物理与光电工程学院舒峰教授工作室
- 本科毕业于华东师范大学
- 博士毕业于中国科学院上海技术物理研究所
- 主持及参与省部级科研项目10余项
- 共发表SCI/EI论文30余篇
- 电话：18106595780
- 邮箱：mzpan@ucas.ac.cn



■ 王义坤（主讲教师）

- 博士，副研究员，电路与系统专业
- 物理与光电工程学院/太极实验室
- 博士毕业于中国科学院上海技术物理研究所
- 主持承担高分专项青年基金等项目，并参与多项国家、省部级项目
- 发表论文10篇，授权专利7项
- 电话：18964590913
- 邮箱：wangyikun@ucas.ac.cn



■ 谢璐璠（学生助教）

- 2023级在读硕士 光电信息工程专业
- 国科大杭州高等研究院 物理与光电工程学院
- 本科毕业于宁波大学
- 电话:15990194631
- 邮箱:787220468@qq.com



■ 李奥（学生助教）

- 2023级在读硕士 光电信息工程专业
- 国科大杭州高等研究院 物理与光电工程学院
- 本科毕业于河南大学
- 电话:15037829584
- 邮箱:1209880960@qq.com



学什么?

怎么学?

学到什么程度?

课程介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

■ 课程目标

- ✓ 学会并熟练掌握一项技能
- ✓ 熟悉基于FPGA的项目开发流程
- ✓ 具备FPGA技术自我进阶学习能力

■ 技能要求

- ✓ 熟悉FPGA原理及内部结构、外围硬件电路设计（供电、配置、时钟、接口....）
- ✓ 熟练掌握FPGA项目的全流程设计方法，具备复杂系统设计与优化的能力
- ✓ 熟练掌握Verilog设计语言，具备“硬件思维”，理解“代码”到“电路”的映射
- ✓ 提升工程实践能力，具备基于FPGA的电路设计及调试能力

学什么？



1. 模拟电路相关知识
2. 数字电路相关知识
3. 程序设计基础

课程前置
基础知识



Practice makes perfect! ! !
Practice makes perfect! ! !
Practice makes perfect! ! !



怎么学?



获得某个领域的专业技能的关键，是以正确的方式练习至少10000小时？
——一万小时定律

正确的方式包括：

1. 明确练习的目标
2. 百分百投入
3. 阶段性总结及分析
4. 走出舒适区



课程介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 课程教学计划

- 第1讲：课程介绍+数字电路基础+工具使用介绍+**Lab0.0: FPGA入门实验 (1)**
- 第2讲：FPGA概述+Vivado工具使用+**Lab0.1: FPGA入门实验 (2)**
- 第3讲：FPGA硬件电路+HDL语言概述+**Lab1: LED控制实验**
- 第4讲：**Verilog语法 (1) + Lab2: 分频计数器**
- 第5讲：**Verilog语法 (2) + Lab3: 流水灯实验**
- 第6讲：**Verilog语法 (3) + Lab4: 数码管实验 (1)**
- 第7讲：状态机+**Lab5: 数码管实验 (2)**
- 第8讲：FPGA时序分析基础+**Lab6: 数码时钟实验**
- 第9讲：课程设计选题+**Lab7: 红绿灯实验**
- 第10讲：**XADC概述及使用+ Lab8: XADC实验**
- 第11讲：**FPGA设计技巧+ Lab9: 串口发送实验**
- 第12讲：**FPGA项目开发及管理**
- 第13讲：课程设计研讨1：项目需求分析讨论
- 第14讲：课程设计研讨2：项目概要设计讨论
- 第15讲：课程设计研讨3：项目详细设计讨论
- 第16讲：课程设计研讨4：板级效果展示+项目总结
- 期末考试：10道题目开卷考试（2小时）

怎么学？

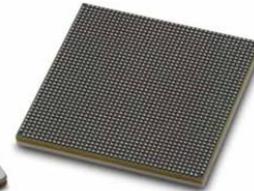
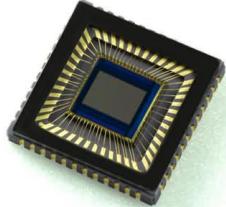
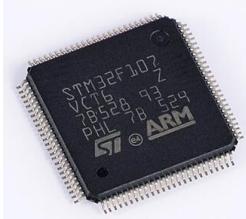




什么是FPGA?

Field Programmable Gate Array

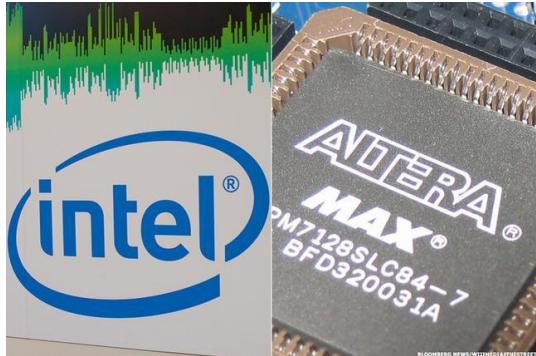
现场可编程（逻辑）门阵列（器件）



课程介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



2015年6月
芯片巨头Intel以167亿美元的价格收购Altera

2020年10月
芯片巨头AMD拟以350亿美元的价格收购Xilinx
2022年2月14日正式完成

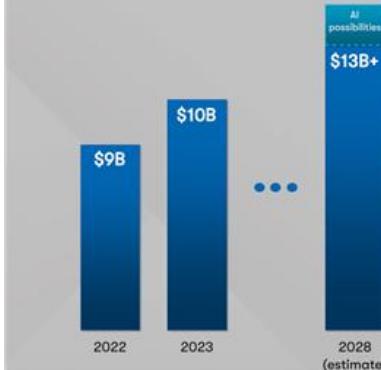
- Intel市值: **1834亿**(2022.2.22), 1030.1亿(2023.2.28), **1928.4亿**(2024.3.4), 984.6亿(2025.3.4)
- AMD市值: **1852亿**(2022.2.22), 1270.1亿(2023.2.28), **3318.2亿**(2024.3.4), 1591.8亿(2025.3.3)
- **2015年6月**, Intel以**每股54美元**的价格收购Altera, 总交易额约为**167亿美元**
- 2020年10月AMD与Xilinx达成协议, 以**全股票交易**方式完成合并交易, Xilinx**每股可换成1.7234股AMD股份**
- 2022年交易完成后, AMD总股数变为16.27亿股, 市值为1852亿美元 (每股113.83美元) , 超越Intel

课程介绍



- 2024年3月1日，芯片巨头英特尔宣布成立全新独立运营的FPGA公司——Altera
- Altera以新形象重回江湖
- 在被Intel收购后，Altera FPGA的重心是高性能的数据中心和云计算，这给了中低端FPGA公司更大的生存空间，宣布独立运营之后Altera又将目光放在了嵌入式和低成本市场，可能给FPGA中低端市场带来变化
- Altera已注意到AI正在为FPGA市场带来巨大机遇，对AI时代FPGA的发展空间充满期待
- Altera将在通信、云、数据中心、嵌入式、工业、汽车等应用领域提供具备AI功能且便于应用的解决方案，Altera的产品组合和路线图将更好地满足云、网络和边缘FPGA市场的需求
- 预计未来五年FPGA的累计潜在市场将超过550亿美元

Our \$55B+ FPGA Opportunity



FPGA market growth across all segments

Expanding Agilex™ FPGA portfolio

AI inferencing for limitless possibilities

Execution Matters Agilex™ Portfolio Announcements



课程介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Products

Applications

Technology

Support



Cloud & Enterprise

AI >

Financial Services >



5G Network Transformation

Wireline >

Wireless >



Edge & Embedded

Automotive >

Consumer Electronics >

Emulation and Prototyping >

Industrial >

Medical >

Military, Aerospace and Government >

Test and Measurement >

AV, Broadcasting, and Video >

■ Intel的FPGA业务应用场景

- 云服务及企业——AI、金融服务
- 5G应用——有线通信、无线通信
- 嵌入式及边缘计算——汽车、消费电子、工业、医疗、国防军工、测试测量

课程介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Artix UltraScale+ FPGA Matches I/O Bandwidth and Compute Density for Next-Generation, Ultra-Compact, Cost-Sensitive Applications



Machine Vision

With high DSP compute density in its class enabled by InFO small form factor packaging, the Artix™ UltraScale+™ FPGA offers high-speed image processing and video pre-processing for factory automation. 12Gb/s and 16Gb/s line rates ensure support for the latest connectivity standards, including 10GigE Vision and CoaXPress 2.1 in the camera or at the frame grabber, as well as PCIe® Gen3 and Gen4* for high speed x86 CPU communication. High performance SLVS-EC v2.0, MIPI, and sub-LVDS support the latest sensor technology, beyond 150-megapixel resolution. A MicroBlaze™ processor in FPGA fabric enables system management for a robust embedded vision camera.

4K UHD Video Converters (SDI to HDMI)

Artix UltraScale+ FPGA in InFO packaging (11.5x9.5mm) can efficiently implement broadcast-quality 4K60 UHD video pipelines for space-constrained applications, such as mini-converters, cameras, and KVM. Massively parallel DSP ensures scalable real-time video processing for multiple AV channels in a single device, and high-performance connectivity enables bridging between SDI, HDMI, DisplayPort, and 1Gb/10Gb Ethernet for live 4K AV-over-IP streaming.



Secure Networking

Artix UltraScale+ FPGAs are a great fit for cost-optimized Nx10G or 25G systems, enabled by 12Gb/s and 16Gb/s transceivers and optimal transceiver count. A common architecture across mid-range and high-end UltraScale+ families allows developers to scale for 100G and 400G systems. The FPGA family is also ideal for bridging for Nx100G systems. Its programmable I/O and PCIe® Gen3/Gen4 provide flexible connectivity and system control across network processing units, and its multi-level security features—including bitstream encryption and DPA counter-measures—provide IP protection and advanced safeguards against network intrusion and cyber attacks.



PON Access



Passive Optical Network (PON) is one of the main broadband access technologies deployed by many operators. Packet processing and traffic management functions are critical to process the aggregated end-user traffic and provisioning per QoS policies. With programmable logic and hardware acceleration blocks, Kintex UltraScale+ FPGAs are well suited to perform Layer 2 to Layer 4 packet processing functions, such as classification, filtering, lookup, and packet forwarding. The integrated 100G Ethernet MAC enables efficient uplink management and saves thousands of LUTs for hardware differentiation. As network operators are always looking for the balance among performance, cost, and power, the Kintex UltraScale+ FPGA's DSP resources, serial connectivity, memory, and logic performance—combined with its power efficiency and ideal price point—makes it suitable for applications such as high-volume 100G PON OLT line cards.

Mobile Backhaul



Wireless traffic has grown tremendously in recent years. The surging demand for higher data capacity leads to innovation in radio access network (RAN) and mobile backhaul applications. Traditional microwave bands support 112MHz signal bandwidth, which requires multiple FPGAs. The Kintex UltraScale+ FPGA is an ideal platform to develop point-to-point microwave modems with higher packet processing in a single device. Increased logic resources, compared to the previous generation, enables end-product differentiation and fast feature deployment in the evolving wireless market. Also, designers can take advantage of up to 20% lower power and easy timing closure enabled by integrated IP.

Data Center Network Acceleration

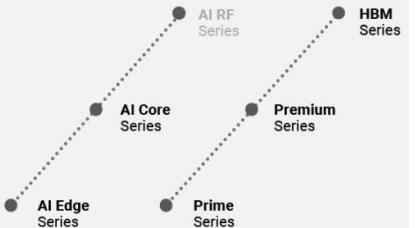


Network port speeds are growing faster than traditional server nodes can keep pace, and new network functions are evolving too quickly for fixed-function devices. Kintex UltraScale+ FPGAs offer an ideal solution to address these challenges, providing high-performance packet processing and datapath offloading, advanced SerDes technology, and 100G Ethernet IP to enable unmatched scalability and connectivity for fast data movement. Kintex UltraScale+ FPGAs deliver a strong balance of performance and power that allows network designers to accelerate a comprehensive range of network workloads while minimizing total cost of ownership. Whether implementing emerging proprietary functionality or accelerating complex algorithms, Kintex UltraScale+ FPGAs provide a compelling solution for the data center with deterministic, ultra low-latency, and adaptable networking solutions with efficient power consumption.

Xilinx FPGA业务应用场景

- 机器视觉、高清转换
- 网络安全、光通信、移动通信、数据中心
- 嵌入式、边缘计算、AI

AMD VERSAL



AI Core Series

Delivers breakthrough AI inference and wireless acceleration with AI Engines that deliver over 100X greater compute performance than today's server-class CPUs.

[View AI Core Series](#)

AI Edge Series

Delivers outstanding AI performance and power efficiency for power- and thermally-constrained edge applications, accelerating the whole application from sensor to AI to real-time control.

[View Edge Series](#)

课程介绍



■ CPU

- 中央处理器。常见的桌面及服务器处理器，核心是**存储程序/数据、串行顺序执行**，通过**多核、多线程**提高数据处理能力

■ GPU

- 图形处理器。早期用于图形显示，逻辑架构与CPU基本一致，**运算单元（ALU）多**，可以提供**并行运算能力**

■ NPU

- 神经网络处理器。针对**神经网络架构**进行了专门优化

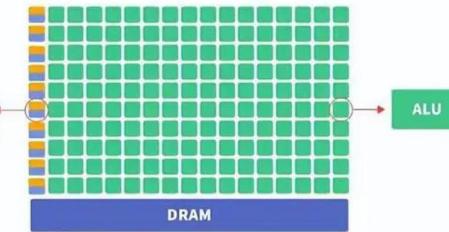
■ FPGA

- 可编程逻辑门阵列。**纯硬件方式**，与**CPU、GPU、NPU**的**指令顺序执行**有明显区别

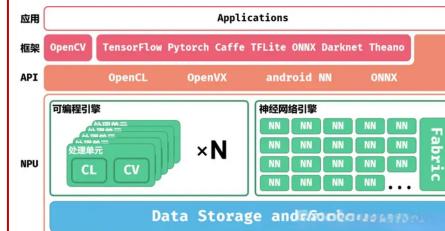
CPUs:Lantency Oriented Design



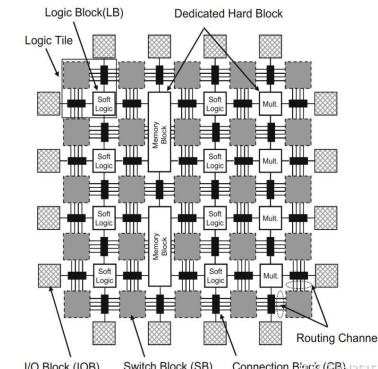
CPU架构



GPU架构



NPU架构



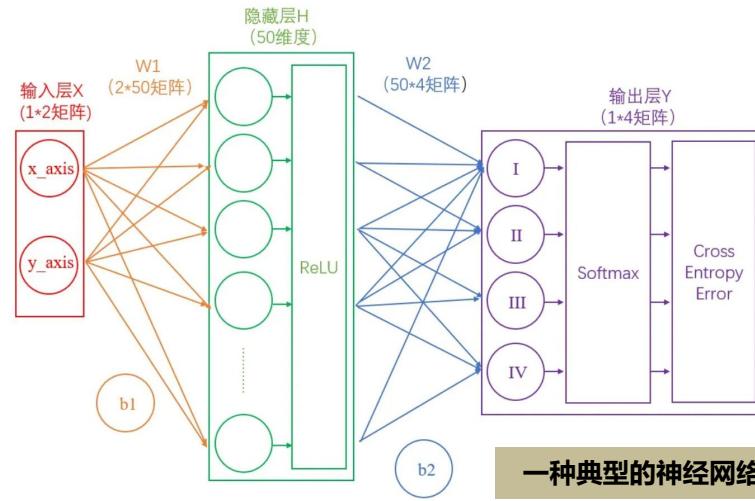
FPGA架构

课程介绍

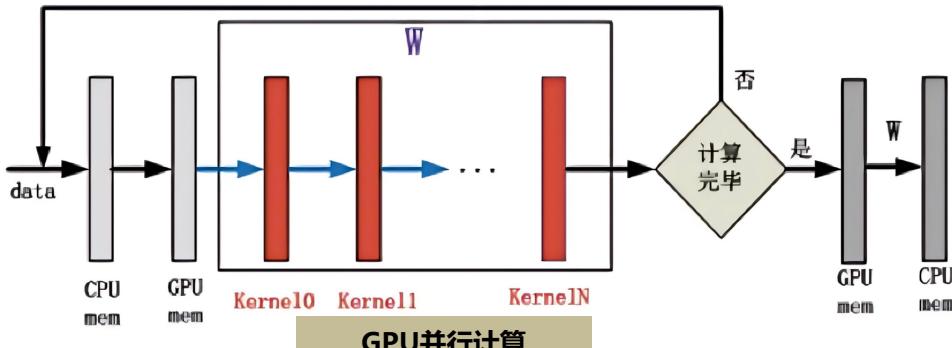


■ 使用GPU、NPU进行深度学习

- 深度学习架构本质上是通过**矩阵运算**构建的，对矩阵运算进行加速能够明显提升架构的运算效率
- CPU**运算时需要取指令进行顺序执行，**核数/线程数有限**，限制了矩阵运算的效率
- GPU、NPU**具有丰富的ALU单元，非常**适合矩阵运算**
- CUDA**: 大规模并发编程模型，使用户可以对NVIDIA GPU进行**并发性编程**
- 深度学习框架**: TensorFlow、Keras、Caffe、PyTorch等



一种典型的神经网络



GPU并行计算

对于CPU，实现数组求和：

```
for(int i=0; i<10; i++)  
    c[i] = a[i] + b[i];
```

大量运行时性能受限

对于GPU，实现数组求和：

```
c[index] = a[index] + b[index];
```

index是GPU的独立线程，实现了并行运算

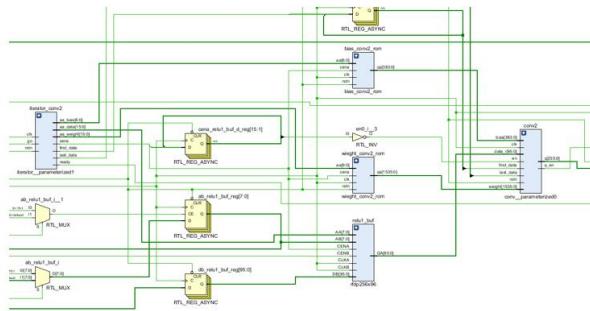
FPGA提供了一种纯硬件方式的计算

课程介绍

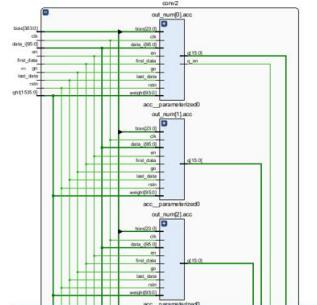


國科大杭州高等研究院 Hangzhou Institute for Advanced Study, UCAS

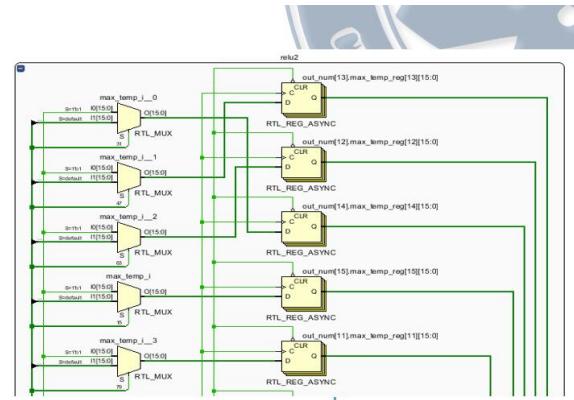
硬件加速：Lenet5卷积网络的FPGA实现



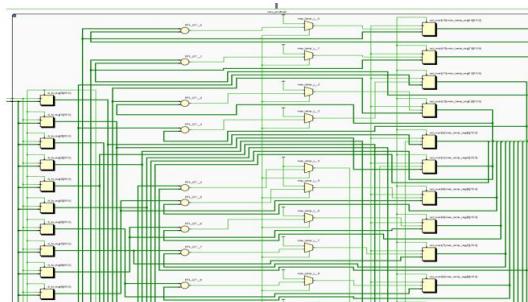
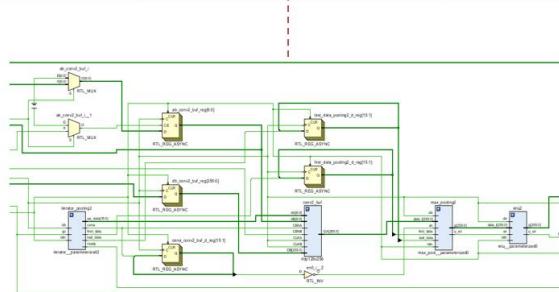
Conv子模块



池化激活实现



池化模块RTL图



课程介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

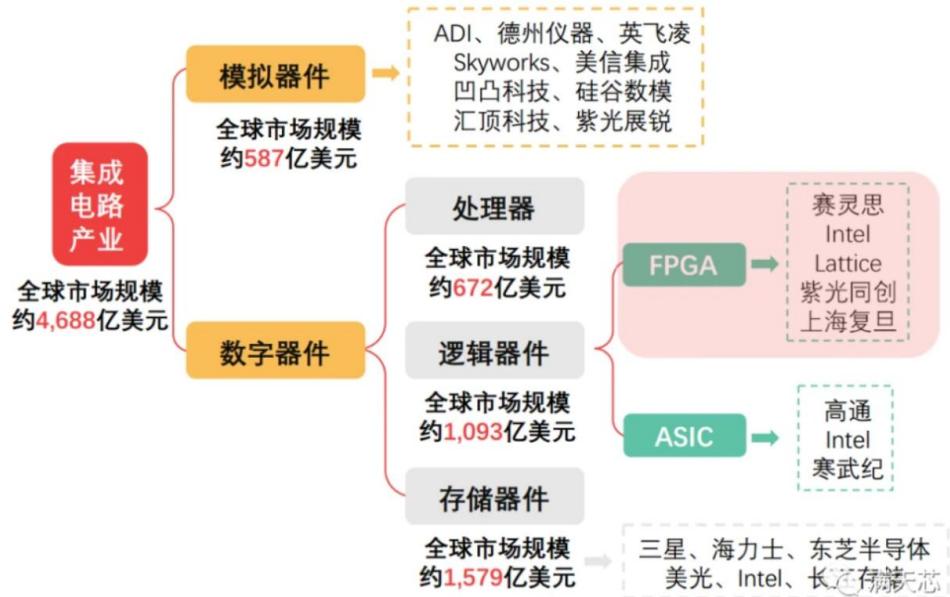
- 在AI大模型领域，FPGA凭借其可定制化、低延迟和高能效等特性，正在成为关键硬件加速方案之一
- 大模型推理加速与低延迟优化：FPGA通过并行计算和流水线架构，显著降低大模型的推理延迟
- 模型压缩与硬件优化：FPGA通过支持低精度计算（如INT8、INT4）和结构化剪枝技术，降低大模型的计算和存储需求；FPGA可针对大模型中的矩阵乘累加（MAC）操作进行硬件级优化，提升矩阵运算效率
- 边缘计算与端侧AI部署：FPGA的低功耗和小体积特性适合部署在智能摄像头、医疗设备等边缘端，直接处理数据并减少云端依赖；FPGA可整合多源传感器数据（如视频、红外等），预处理后输入大模型
- 软硬件协同与开发工具革新：通过OpenCL、HLS等工具，开发者无需硬件描述语言即可将AI模型转换为FPGA硬件代码
- 异构计算协同与扩展性：FPGA可作为AI集群中的预处理单元或网络接口卡（NIC），减少GPU的数据摄入抖动；随着大模型参数规模增长，FPGA的灵活重构能力可适应算法迭代

FPGA在AI大模型中的核心价值在于灵活定制硬件架构与边缘端高效推理

尤其在需要低延迟、高能效和实时处理的场景中不可替代

随着开发工具和异构计算生态的成熟，有望成为大模型从云端到边缘端全链路加速的关键支撑

课程介绍



2016–2025年全球FPGA芯片市场规模及增速



- 2021年全球芯片销售额首次突破5000亿美元，2024年达到6276亿美元
- FPGA市场预测到2025年达到125亿美元，年复合增长率10.22% (华经产业研究院)
- 全球FPGA市场由四大巨头Xilinx, Intel, Lattice, Microsemi垄断
- 硬件设计和高端的EDA软件设计上都形成了极强的技术封锁

课程介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



- 整个FPGA产业链包括上游、中游和下游
- 上游：IP设计、晶圆代工、EDA软件
- 中游：FPGA芯片
- 下游：应用端
- FPGA芯片方面Xilinx和Altera市场占有率达到87%，Xilinx 56%，Altera 31%

课程介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

■ 紫光同创

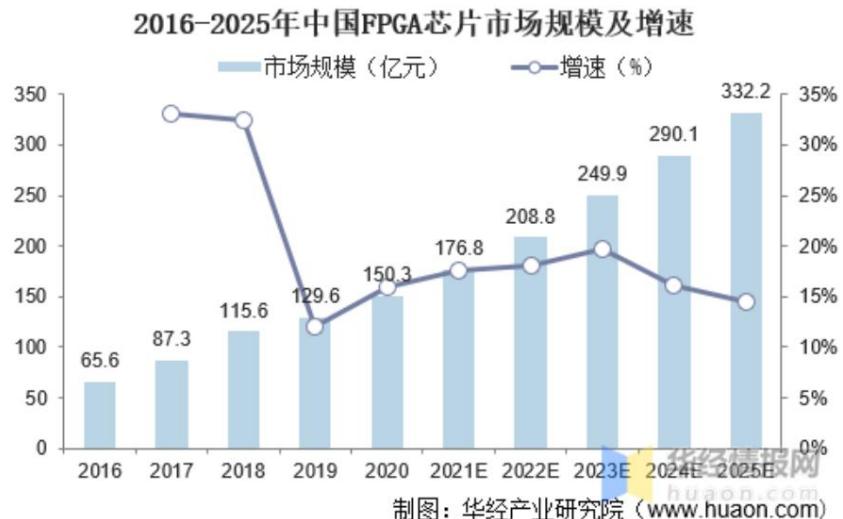
- ✓ Titan、Logos、Compa系列FPGA, Pango Design Suite设计工具
- ✓ 自主知识产权的可编程逻辑器件平台和系统解决方案
- ✓ 产品覆盖通信、工业控制、视频监控、消费电子、数据中心等应用领域

■ 安路科技

- ✓ Saleagle、Salelf系列FPGA, TangDynasty设计工具
- ✓ 产品应用于工业控制、网络通信、消费电子、数据中心等领域
- ✓ 2021年底科创板上市，市值最高超过300亿，现市值128.6亿元人民币

■ 复旦微电子

- ✓ 产品用于卫星导航、航空航天等领域



- **中国市场：2020年150亿元，预测到2025年超过300亿元**
- **国内FPGA公司：紫光同创、安路科技、复旦微电子、京微齐力、上海遨格芯、广东高云、西安智多晶**
- **FPGA行业技术壁垒高：硬件开发部分属于典型的 IC 设计，需要配套 EDA 软件一起使用**

课程介绍



• 学习资源

• 参考书籍

Xilinx Artix-7 FPGA 快速入门、技巧及实例，吴厚航

Verilog数字系统设计教程，夏宇闻

• Verilog语法学习

HDLbits

语法工具书

• Vivado工具使用

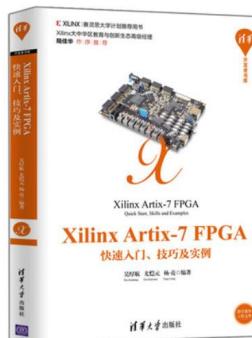
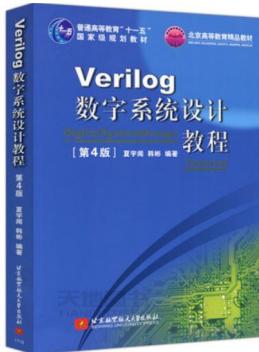
Vivado帮助文件

• 硬件学习资源

Xilinx硬件手册

EGo1口袋实验室

物光学院/太极实验室电子学硬件资源



怎么学？

[Vivado : User Guides](#)

- [Vivado Design Suite User Guide: Getting Started](#)
- [Vivado Design Suite User Guide: Design Flows Overview](#)
- [Vivado Design Suite User Guide: Using the Vivado IDE](#)
- [Vivado Design Suite User Guide: Using Tcl Scripting](#)
- [Vivado Design Suite User Guide: System-Level Design Entry](#)
- [Vivado Design Suite User Guide: Designing with IP](#)
- [Vivado Design Suite User Guide: Creating and Packaging Custom IP](#)
- [Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator](#)
- [Vivado Design Suite User Guide: Embedded Processor Hardware Design](#)
- [Vivado Design Suite User Guide: Model-Based DSP Design using System Generator](#)
- [Model Composer User Guide](#)
- [Vivado Design Suite User Guide: High-Level Synthesis \(HLS\)](#)
- [Vivado Design Suite User Guide: I/O and Clock Planning](#)
- [Vivado Design Suite User Guide: Logic Simulation](#)
- [Xilinx Power Estimator User Guide](#)
- [Vivado Design Suite User Guide: Using Constraints](#)
- [Vivado Design Suite User Guide: Synthesis](#)
- [Vivado Design Suite User Guide: Implementation](#)
- [Vivado Design Suite User Guide: Hierarchical Design](#)
- [Vivado Design Suite User Guide: Partial Reconfiguration](#)
- [Vivado Design Suite User Guide: Power Analysis and Optimization](#)
- [Vivado Design Suite User Guide: Design Analysis and Closure Techniques](#)
- [Vivado Design Suite User Guide: Programming and Debugging](#)
- [Vivado Isolation Verifier User Guide](#)
- [Vivado Isolation Verifier User Guide \(Tcl Based\)](#)
- [Bootgen User Guide](#)

课程介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

hdldbts.01xz.net/wiki/Main_Page

新标签页 HHIT-NAS HOME-NAS FPGALab-NAS 统一身份认证平台 光电成像技术 杭州政府

HDLBits Problem Set Simulation My Profile Help 01xz.net

HDLBits — Verilog Practice

HDLBits is a collection of small circuit design exercises for practicing digital hardware design using Verilog Hardware Description Language (HDL). Earlier problems follow a tutorial style, while later problems will increasingly challenge your circuit design skills.

Each problem requires you to design a small circuit in Verilog. HDLBits gives you immediate feedback on the circuit module you submit. Your circuit is checked for correctness by simulating with a set of test vectors and comparing it to our reference solution.

How to use HDLBits

1. Choose a problem: [Browse the problem set](#) or [go to the first problem](#)
2. Write a solution in Verilog
3. Submit, simulate, and debug if necessary

If you want to [track your progress](#) or move to another browser, [create a username and password](#) so you can [log in](#) from elsewhere.

Problem Stats

Problem	Success	Incorrect	Compile error	Simulation error	Total attempts	Success rate (%)
1 step_one	133681	8592	152886	5210	301788	44%
2 zero	113114	3550	77015	1043	195031	58%
3 wire	121174	17407	110038	2907	252385	48%
4 wire4	112820	9398	52035	333	174785	65%
5 notgate	110450	5818	37969	679	155063	71%
6 andgate	111647	11717	24540	359	148445	75%
7 norgate	105427	69575	49830	213	225197	47%
8 xnorgate	104684	53122	33211	92	191228	55%
9 wire_decl	103665	27792	47014	412	179049	58%
10 7458	97129	21570	36570	153	155639	62%
11 vector0	97515	12637	43831	705	154972	63%

Contents [hide]

1 Getting Started	
2 Verilog Language	
2.1 Basics	
2.2 Vectors	
2.3 Modules: Hierarchy	
2.4 Procedures	
2.5 More Verilog Features	
3 Circuits	
3.1 Combinational Logic	
3.1.1 Basic Gates	
3.1.2 Multiplexers	
3.1.3 Arithmetic Circuits	
3.1.4 Karnaugh Map to Circuit	
3.2 Sequential Logic	
3.2.1 Latches and Flip-Flops	
3.2.2 Counters	
3.2.3 Shift Registers	
3.2.4 More Circuits	
3.2.5 Finite State Machines	
3.3 Building Larger Circuits	

173 sim/circuit10	7642	9439	3362	114	20572	37%
174 tb/clock	18160	14879	33516	971	67683	27%
175 tb/tb1	14616	9588	9381	687	34317	43%
176 tb/and	14133	8639	25761	1305	49872	28%
177 tb/tb2	12813	11075	16709	645	41269	31%
178 tb/tff	13117	12264	10550	399	36361	36%
179 cs450/timer	3831	11714	5206	189	20991	18%
180 cs450/counter_2bc	2966	3702	3434	86	10213	29%
181 cs450/history_shift	2240	2524	1548	25	6339	35%
182 cs450/gshare	1932	5945	3042	67	10987	18%
Attempts:	6620432	4633027	6040177	103270	17414671	38%
Problems:	182				182	100%

课程介绍



國科大杭州高華研究院 Hangzhou Institute for Advanced Study, UCAS

The image shows the landing page for DeepSeek-R1. At the top, there is a banner with the text "DeepSeek-R1 已发布并开源, 性能对标 OpenAI o1 正式版, 在网页端、APP 和 API 全面上线, 点击查看详情." Below the banner, the DeepSeek logo is displayed in large blue letters, followed by the tagline "探索未至之境". The page is divided into two main sections: "开始对话" on the left and "获取手机 App" on the right, each with its own descriptive text and icons.

The screenshot shows the MarsCode AI interface. The top menu bar includes File (文件), Edit (编辑), Select (选择), View (查看), Go to (转到), Run (运行), Terminal (终端), and Help (帮助). The title bar says MARS CODE AI. On the left, there are icons for file operations like Open, Save, and Find. A user profile is shown with the name "用户91279197740". A tooltip says "使用Verilog编写一个状态机". In the center, there's a project section titled "MarsCode AI Doubao-1.5-pro". Below it, a text area contains a Verilog code snippet for a sequence detector. The code defines a module named "sequence_detector" with inputs for clock (clk), reset (rst_n), and input signal (in), and an output for detection (detect). It uses a state register (reg) to detect the sequence. The code is syntax-highlighted, and there are buttons for Apply and Run.

```
module sequence_detector (
    input wire clk,           // 时钟信号
    input wire rst_n,         // 异步复位信号, 低电平有效
    input wire in,            // 输入信号
    output reg detect         // 检测到序列时输出高电平
);

```

A screenshot of a Chinese AI search interface. At the top, there's a navigation bar with icons for home, search, and other functions. Below it is a sidebar with various AI features like search, writing, image generation, reading, programming, voice calls, website viewing, and intelligent bodies, each with a small icon and a red-bordered box around the first few. The main area has a large blue button labeled '新手任务' (Newbie Task). A central message box displays the text '早上好, mzpan' (Good morning, mzpan) in large white font. Below the message is a input field with placeholder text '发消息、输入 @ 或 / 选择技能'. At the bottom are several small icons for phone, file, and other functions.

课程介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 基本要求 (入门要求)

- ✓ 了解FPGA相关技术，包括基础理论，芯片架构，硬件电路，开发流程
- ✓ 学会Verilog设计语言，熟练使用EDA工具完成FPGA代码设计及仿真
- ✓ 能够在FPGA开发板上设计并实现基础的硬件电路功能

■ 进阶要求 (熟练要求)

- ✓ 用合适的代码风格设计出最优的电路
- ✓ 熟练使用EDA工具对设计进行优化
- ✓ 掌握板级电路调试的技巧，具备基于FPGA的项目开发能力

学到什么程度？

课程介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

学会此项技能可以做什么？

0. 进入课题组参与科研项目的一项基础技能

1. 科研院所、军工企业有大量的FPGA相关岗位需求

2. 芯片设计（前端、仿真、测试等）

3. 通信、汽车、安防类企业

4. 人工智能、硬件加速、算法加速等新兴行业

□ FPGA工程师 3.5-7万·16薪

fpga 五险一金 员工旅游 专业培训 年终奖金

 上海瑾讯微信息技术有限公司 电子技术/半导体/集成电路 民营 150-500人 [立即沟通](#) 1天内处理简历

□ FPGA原型验证工程师 3.5-6.5万

测试 测试用例 调试 C++ 软件测试 fpga 英语读写 测试流程 硬件 接口 餐饮补贴 年终奖金

 进迭时空（杭州）科技有限公司 电子技术/半导体/集成电路 民营 50-150人 [立即沟通](#)

□ 高级硬件研发工程师——FPGA（杭州） 2.5-4.5万

弹性工作 扁平管理 晋升空间大 用户过亿 大牛带队 免费三餐

 字节跳动 互联网/电子商务 民营 10000人以上 [立即沟通](#)

□ FPGA主管工程师 2.5-4万·15薪

xilinx altera 五险一金 补充医疗保险 员工旅游 专业培训 年终奖金 定期体检 免费工作餐

 杭州长川科技股份有限公司 电子技术/半导体/集成电路 已上市 1000-5000人 [立即沟通](#) 刚刚活跃

□ FPGA工程师(J10296) 2.4万·14薪

餐饮补贴 交通补贴 年终奖金 员工旅游 带薪年假 五险一金

 先临三维科技股份有限公司 仪器仪表/工业自动化 已上市 1000-5000人 [立即沟通](#) 简历处理快

□ FPGA工程师 2.4万·15薪

计算机 电子 需求分析 通信 代码编写 eda 架构设计 fpga 芯片测试 仿真 五险一金 餐饮补贴

 思看科技（杭州）股份有限公司 仪器仪表/工业自动化 民营 150-500人 [立即沟通](#)

课程介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

FPGA工程师 上海瑾讯微信息技术有限公司 3.5-7万·16薪

杭州 | 5-7年 | 本科

五险一金 员工旅游 专业培训 年终奖金

职位信息

工作职责:

- 1、负责FPGA设计开发工作；
- 2、参与硬件验证平台设计，FPGA硬件选型，平台开发和调试工作；
- 3、协助数字前端设计人员完成原型验证；
- 4、芯片debug测试和问题反馈。

任职要求:

- 1、电子工程类等相关专业本科及以上学历，2年及以上工作经验；
- 2、精通Verilog语言，精通主流公司FPGA开发；
- 3、具有大规模FPGA项目开发经验者优先；
- 4、具有SoC架构FPGA设计经验者优先；
- 5、具有板级硬件设计能力者优先；
- 6、具有图像处理、视频算法FPGA实现经验者优先；
- 7、具有AI相关应用FPGA开发经验者优先；
- 8、具有较强的学习能力，独立解决问题能力和良好的团队合作精神。

职能类别: FPGA开发工程师

高级硬件研发工程师——FPGA（杭... 2.5-4.5万

杭州 | 3-4年 | 本科

弹性工作 扁平管理 薪升空间大 用户过亿 大牛带队 免费三餐

字节跳动

职位信息

职位描述:

- 面向数据中心网络和存储等相关业务的FPGA加速系统设计和实现
- 1、根据需求，进行FPGA方案选型和设计；
 - 2、负责FPGA逻辑设计、仿真和调试；
 - 3、负责FPGA加速产品上线后的自动化运维；
 - 4、协助板级硬件工程师设计、开发和调试FPGA板卡。

职位要求:

- 1、熟悉主流厂商的FPGA芯片和工具，深入理解现代FPGA基本原理和架构；
- 2、精通SystemVerilog或Verilog语言，熟练掌握FPGA开发、验证和时序优化技巧；
- 3、有网络或存储硬件加速相关的项目经验；
- 4、聪明好学，积极主动，有自驱力。

加分项:

- 1、有智能网卡设计、开发、验证和运维经验；
- 2、熟悉FPGA常用的IO协议和相关控制器IP，如PCIe、DDR、Ethernet等。



成绩计算

20% (Lab1~Lab10) + 50% (课程设计) + 30% (期末考试)

课程设计

锻炼团队协作能力，理解项目开发流程

鼓励自由选题（建议与FPGA类竞赛结合）

3人一组，自由组队

课程介绍



國科大杭州高華研究院 Hangzhou Institute for Advanced Study, UCAS

The screenshot shows the homepage of the 9th China College IC Competition. At the top, there's a navigation bar with links to HOME-NAS, FPGALab-NAS, a unified certification platform, lighting technology, Hangzhou Government, and Verilog. The main header features the competition's logo (a blue and red geometric cube) and the text '全国大学生集成电路创新创业大赛 CHINA COLLEGE IC COMPETITION'. Below the header, there are five tabs: 关于大赛 (About the Competition), 主赛道参赛 (Competing in the Main Track), 职业技能赛道 (Competing in the Vocational Skills Track), 新闻公告 (News Announcements), and 合作单位 (Partners). The central part of the page has a large banner with the text '一起从芯出发' (Starting from the Chip) and '第九届全国大学生集成电路创新创业大赛 正式启动' (The 9th China College IC Competition officially starts). At the bottom, there's a footer with logos of various sponsors and partners.



课程介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

1.这门课程我最喜欢什么?

1. 这门课程我最喜欢授课的老师、助教学长、授课内容、学到的东西
2. 课程结合项目开发流程，理论与实践结合。
3. 老师讲课风趣幽默，讲授内容详实有用
4. 试验实践和项目实践，以及老师细致的讲解
5. 老师的讲解生动活泼，能够很好的带动学生们的积极性
6. 这个课太难了，老师非让我选
7. 所学知识的实用性，老师认真负责的教学态度。
8. 老师讲课干货很多，学习到了很多重要的知识，非常好！！！
9. 实验教程讲解详细，适合无基础人
10. 第一部分的基础知识讲的非常快，后面的都很详细，最喜欢的就就是数量合理的作业，加深了对基础概念的理解
11. 在课堂上可以学到很多有用的东西，老师讲的很清晰，让我受益匪浅！
12. 内容充实，生动有趣，作业合理，收获颇丰
13. 这门课程我最喜欢的是小组作业部分
14. 老师讲的很有意思，结合生活，生动形象
15. 11111111111111111111111111111111
16. 这门课课程我最喜欢的是老师严谨的教学风格。
17. 这门课程我最喜欢老师教授的应用技能
18. 授课老师讲课风趣，能结合时事进行授课
19. 好
20. 11111111111111111111111111111111
21. 11111111111111111111111111111111
22. 老师上的内容非常实用，对我后续科研开展大有裨益。
23. 老师讲课方式和课程内容
24. 老师讲课非常好，能学到非常多有用的知识
25. 这门课我最喜欢老师讲授的一些很实用或者很新颖的知识点
26. 这门课程我最喜欢老师讲课侃侃而谈
27. 教材、参考书等阅读资料对学习有帮助

3.我平均每周在这门课程上花费多少小时?

1. 我平均每周在这门课上花费10个小时
2. 约15小时。
3. 我平均每周在这门课程上花费5——8个小时
4. 20小时以上
5. 我平均每周在这门课程上花费7小时的时间
6. 1h
7. 我平均每周在这门课上花费20多个小时。
8. 老师讲的很有意思，结合生活，生动形象
9. 66666666666666666666666666
10. 我平均每周在这门课程上花费8、9个小时
11. 每周可以花费大概4个小时，不固定，有时多有时少。
12. 内容充实，生动有趣，作业合理，收获颇丰
13. 我平均每周在这门课程上花费8小时
14. 我平均每周在这门课程上花费0.5小时
15. 11111111111111111111111111111111
16. 我平均每周在这门课程上花费10个小时左右。
17. 我平均每周在这门课程上花费10个小时
18. 平均每周有三节课，并且有一次课后作业，一般需要花费4个小时时间

课程介绍



2. 我认为本课程应从哪些方面需要进一步改进和提高？

1. 我认为本课程可以从代码的难度递增上改进一下，感觉到后面没点基础就不太跟得上了

2. 无.....

3. 希望能更加与现阶段工作相结合。

4. 课后需要花很多时间钻研，希望占的学分更多一些

5. 可能需要增加一点点老师和同学的互动，便于大家更好的理解老师讲授的知识

6. 最后的课程报告完善一下吧 后期其他考试都赶一起 都忙不过来了

7. 就目前的学习来看，本课程进度安排合理，内容充实，无需改进。

8. 没有什么意见或建议，老师的课非常棒！！！

9. 没有没有没有没有没有没有没有没有

10. 第一部分的基础知识讲的非常快，如果再详细点就更好了

11. 课堂互动再多一点，必要的时候可以停一下，与学生互动要增多！

12. 内容充实，生动有趣，作业合理，收获颇丰

13. 我认为这门课程没有什么需要改进的

14. 我认为本课程不需要进一步改进和提高

15. 11111111111111111111111111111111

16. 我认为本课程没有什么需要进一步改进和提高。

17. 我认为本课程没有哪些方面需要进一步改进和提高

18. 可以通过邀请学生上台演讲，使学生有更高的参与度，更好地融入到课堂中

19. 好

20. 11111111111111111111111111111111

21. 11111111111111111111111111111111

22. 都很好

23. 暂时没有

24. 老师讲课非常好，能学到非常多有用的知识

25. 我认为这门课讲得挺好的，学到的相关理论很受用

26. 我认为本课程应从课程设置、作业布置进一步改进和提高

27. 课程的任务量（包括作业、测验、测试、报告、论文等）进一步增加

28. 无

29. 感觉老师可以给一些错误的代码，这样大家就不敢直接抄了（当然，希望module定义的部分不要给错误的东西）

30. 不用改进.....

31. 对传统教学模式进行了大胆探索，以小组合作的形式充分发挥了学生的主观能动性，剩下的问题都是我的。

32. 我学习到解决实际问题的方法 我发展了原创性研究所需的技能

33. 可以考虑加入一些文献阅读，让学生了解行业动态。

34. 各方面都很好，我觉得不需要提高了

35. 无

36. 课堂互动再多一点，必要的时候可以停一下，与学生互动要增多！

37.

38. 课程量上应该进一步增加，可以更加深入的去学习



THANKS



國科大杭州高等研究院
Hangzhou Institute for Advanced Study, UCAS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第一讲 (1.2 数字电路基础)

2025年3月4日

数字&模拟



■ 什么是信号?

- **定义:** 信号是运载消息的工具, 是消息的载体。
- **分类:** 对信号的分类方法很多, 可以分为确定性信号和随机信号、模拟信号和数字信号、能量信号和功率信号、时域信号和频域信号、时限信号和频限信号、实信号和复信号等。

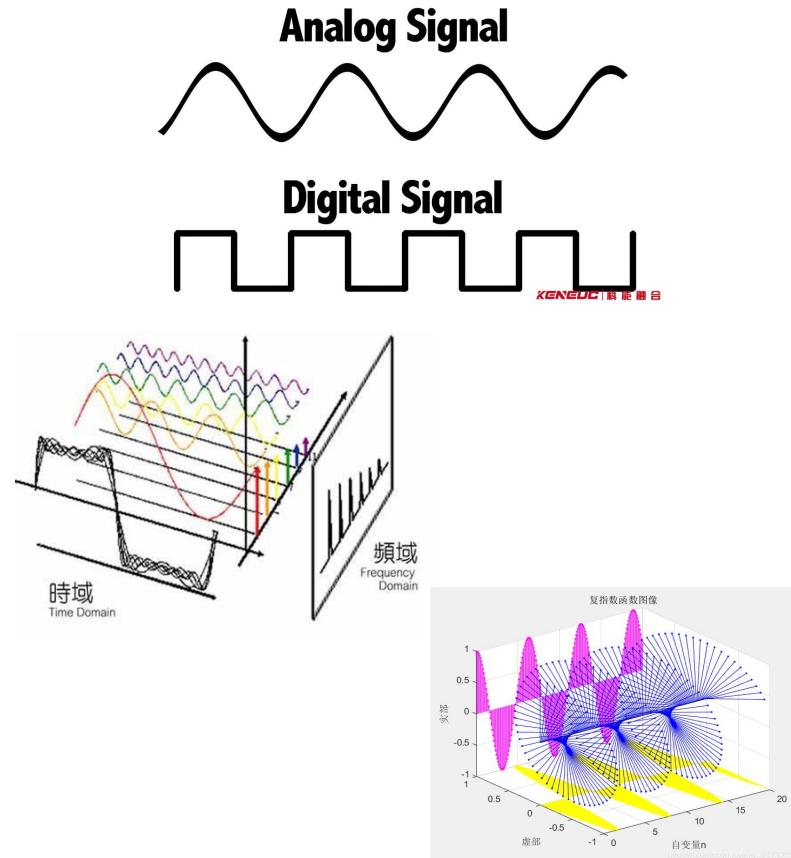
■ 模拟信号

- **定义:** 模拟信号是指信号波形模拟着信息的变化而变化, 其主要特征是幅度是连续的, 可取无限多个值; 而在时间上则可连续, 也可不连续。

■ 数字信号

- **定义:** 数字信号是指不仅在时间上是离散的, 而且在幅度上也是离散的, 只能取有限个数值的信号。
- **数字信号优势:**

- a) **抗干扰性强:** 数字信号不容易受到噪声的影响, 更加可靠。
- b) **易于存储和处理:** 数字信号可以方便地存储在存储器中, 并且可以通过各种算法进行处理。
- c) **可编程性:** 可以通过改变程序来改变数字电路的功能, 更加灵活。

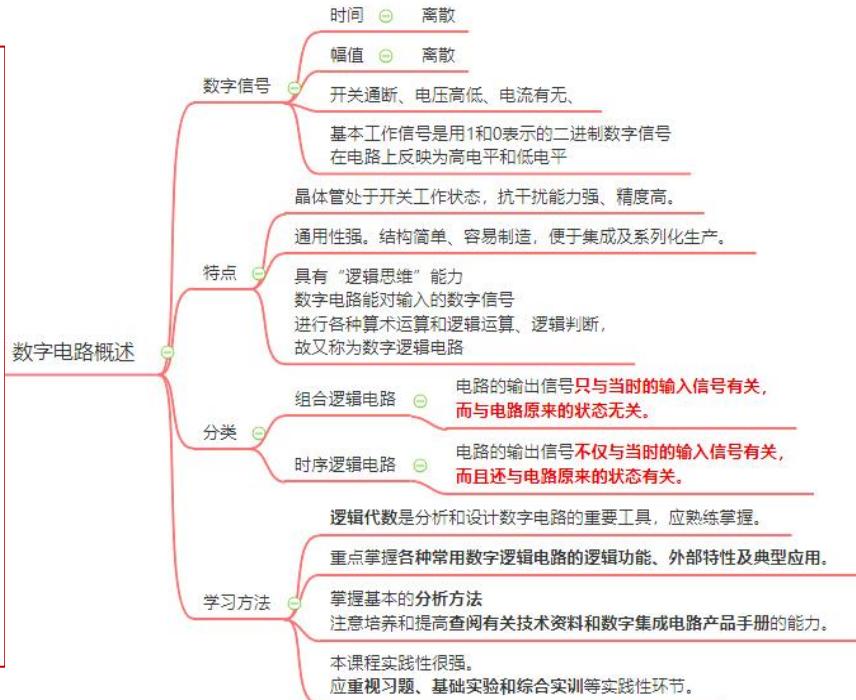


数字电路课程



■ 数字电路课程的主要内容

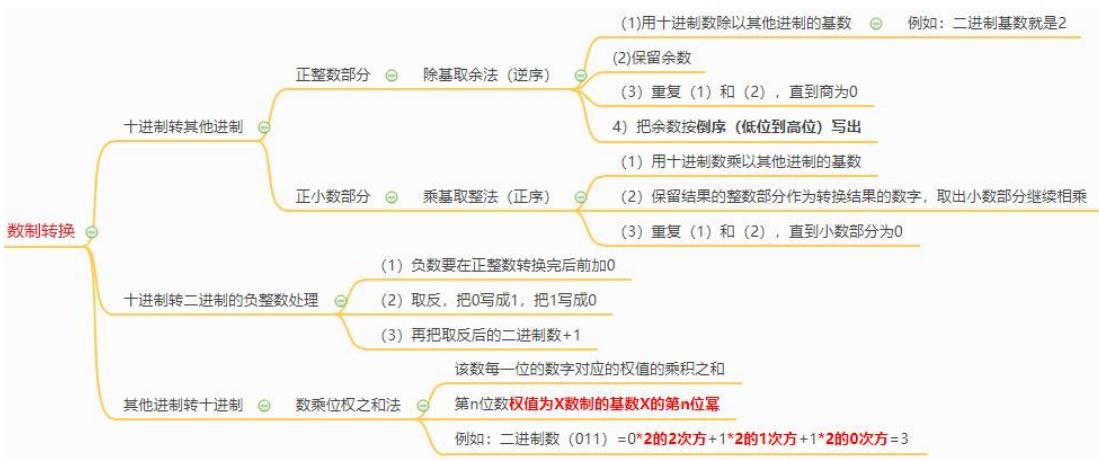
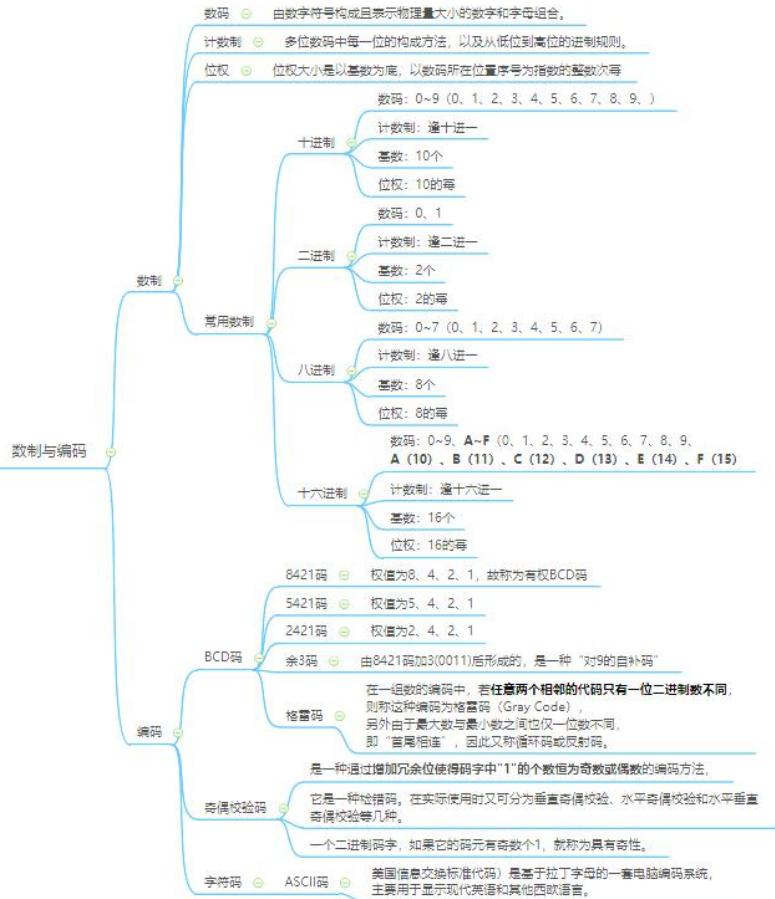
- **数制:** 二进制、八进制、十进制、十六进制、BCD码、格雷码、ASCII码
- **逻辑代数:** 基本定律（结合律、交换律、吸收率等），代数规则（代入、反演、对偶），逻辑化简，卡诺图
- **门电路:** 与门、或门、非门、异或门、同或门，CMOS、TTL
- **组合逻辑电路:** 编码器、译码器、数据选择器、竞争、冒险
- **锁存器及触发器:** 双稳态、锁存器、D触发器、JK触发器、T触发器、SR触发器
- **时序逻辑电路:** 同步时序、异步时序、时序图、状态图、移位寄存器、计数器
- **存储器、CPLD、FPGA:** ROM、SDRAM、CPLD、FPGA
- **脉冲波形的变换与产生:** 单稳态触发器、施密特触发器、多谐振荡器、555定时器
- **数模和模数转换器:** D/A转换器、A/D转换器
- **数字系统:** 数字系统组成，设计方法，实现，状态机



数制、编码、数制转换



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS



逻辑运算，门电路

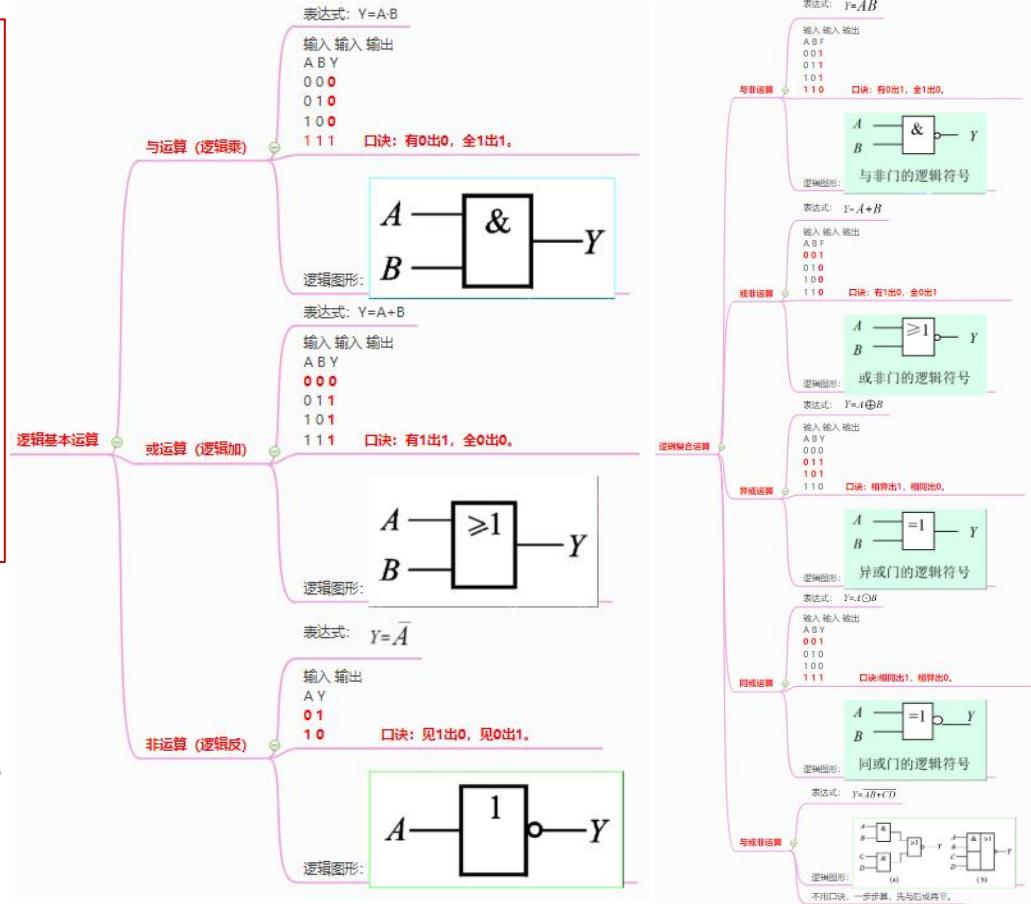
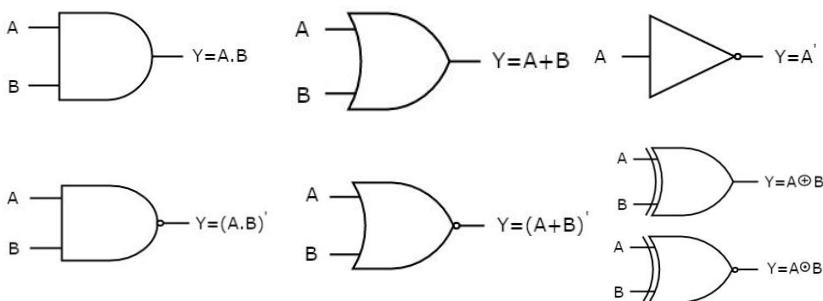


■ 逻辑运算

- 基本逻辑运算：与、或、非
- 复合逻辑运算：与非、或非、异或、同或、与或非
- 逻辑运算基本定律：结合律、交换律、分配律、反演率、吸收率
- 逻辑代数基本规则：代入规则、反演规则、对偶规则
- 逻辑化简：与或表达式，逻辑化简方法，卡诺图

■ 门电路

- 基本门电路：与门、或门、非门、异或门、同或门
- 门电路特性：高低电平、噪声容限、传输延迟时间、扇入扇出
- 电路形式：CMOS、NMOS、TTL、ECL、FET



组合逻辑电路 (Combinational Logic)



國科大杭州高等研究院

Hangzhou Institute for Advanced Study, UCAS

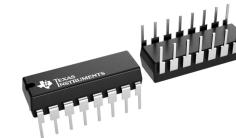
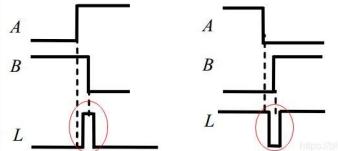


组合逻辑电路 竞争与冒险

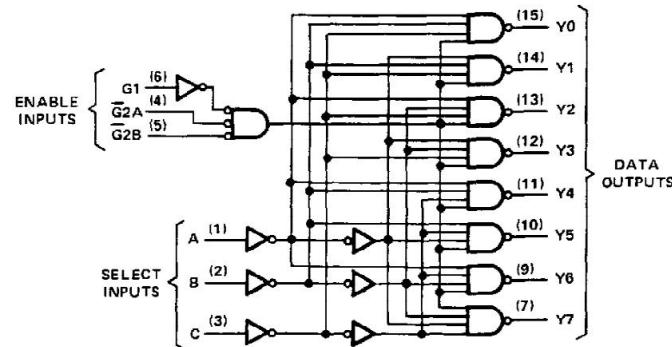
$$L \equiv AB \equiv 0 \quad L \equiv A + B \equiv 1$$



考虑门的延时时间，且用非门实现 $B = \overline{A}$ 时



'LS138, SN54S138, SN74S138A



**'LS138, SN54138, SN74S138A
FUNCTION TABLE**

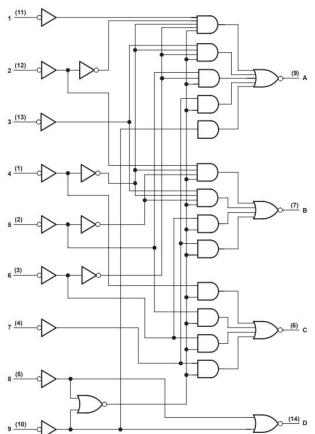
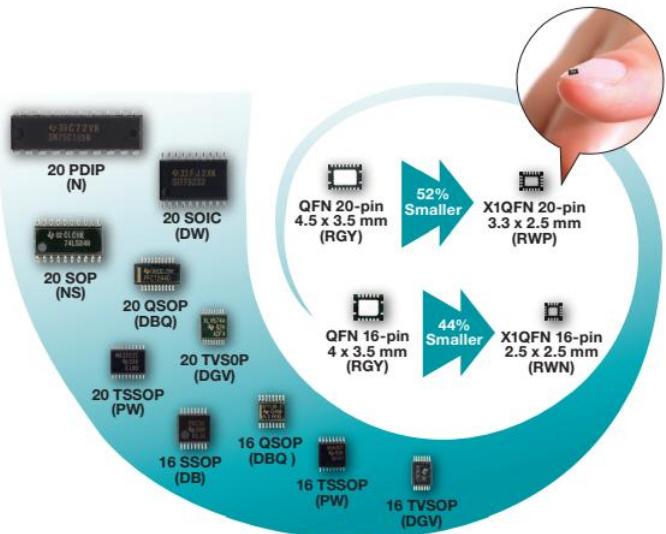
组合逻辑电路 (Combinational Logic)



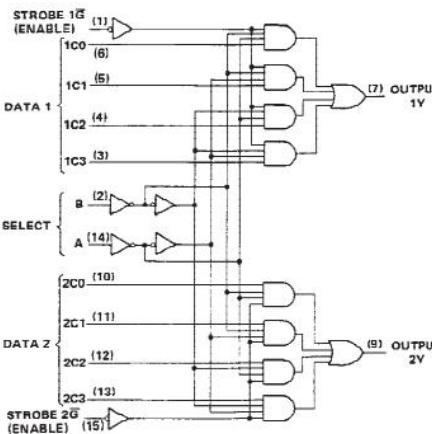
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 常见组合逻辑电路

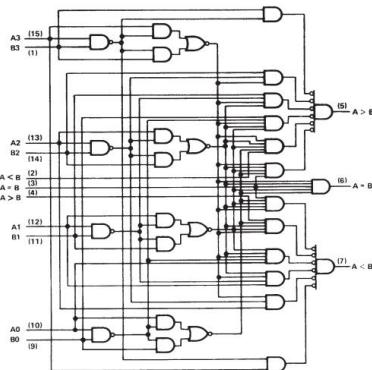
- 译码器: 74138、74142、74144、74144、74237
- 编码器: 74147、74148、74149
- 多路复用器: 74151、74153、74251、74257
- 数字比较器: 7485、4063、74683
- 算数单元: 74181、74283、74280



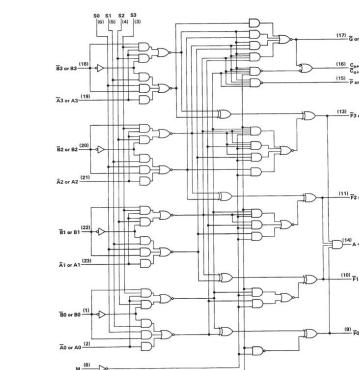
74LS147



74LS153

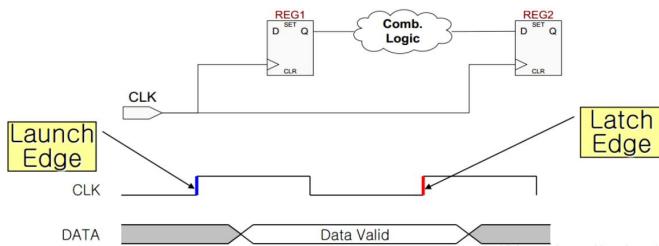


74LS85



74LS181

时序逻辑电路 (Sequential Logic)



■ 建立时间 (Tsu)

- 采样时刻（时钟上升沿）之前，数据需要稳定的最短时间，为建立时间

■ 保持时间 (Th)

- 采样时刻（时钟上升沿）之后，数据需要维持的最短时间，为保持时间

■ 亚稳态

- 指触发器在某一段时间内不能达到一个确定的状态
- 触发器的输出端Q在较长时间内处于振荡状态，不等于输入端D
- 振荡状态的时间称作决断时间 (resolution time)
- 亚稳态出现的主要原因是触发器无法满足建立时间或者保持时间

■ 亚稳态的传播

- 亚稳态震荡会导致传播到下一级，引起更大范围的故障
- 改善方法：降低时钟频率，用更快的触发器，两级同步，改善时钟质量

时序电路在任何时刻的稳定输出，不仅与该时刻的输入信号有关，而且与电路原来的状态有关。

为了记忆电路的状态，时序电路必须包含有存储电路。存储电路通常以触发器为基本单元电路构成。

时序电路的逻辑功能可用逻辑表达式、状态表、卡诺图、状态图、时序图（波形图）和逻辑图6种方式表示，这些表示方法在本质上是相同的，可以互相转换。

状态表跟真值表差不多，只是分开了现态和次态，根据现态（最小项）求出对应次态（结果）

画出状态现态从0开始一直走向次态的关系图可以发现它们之间的关系，例如构成了循环。

各个触发器的时钟脉冲相同，即电路中有一个统一的时钟脉冲，每来一个时钟脉冲，电路的状态只改变一次。

各个触发器的时钟脉冲不同，即电路中没有统一的时钟脉冲来控制电路状态的变化，电路状态改变时，电路中要更新状态的触发器的翻转有先有后，是异步进行的。

输出不仅与现态有关，而且还决定于电路当前的输入；其输出仅决定于电路的现态，与电路当前的输入无关；或者根本就不存在独立设置的输出，而以电路的状态直接作为输出。

电路图→时钟方程、驱动方程和输出方程→状态方程→计算→状态图、状态表或时序图→判断电路逻辑功能

时钟方程: $CP = ?$ (同步时序电路的时钟方程可省去不写。)

输出方程: $Y = ?$ (输出仅与电路现态有关，为穆尔型时序电路。)

驱动方程: $R = ? / S = ? / D = ? / J = ? / K = ? / T = ?$

状态方程: $Q = ?$ 驱动方程代入各触发器的特性方程，得到状态方程。

左边: 输入、现态。

右边: 次态、输出。

一般从现态为0开始计算下一个次态，观察从图的走向规律，比如是不是循环，是加法方向还是减法方向。

时序图也就是波形图更直观看出在时间轴上各个量随着CP的改变，然后根据这些猜出该电路的具体功能

在任何时刻的输出不仅和输入有关，而且还决定于电路原来的状态。

特点



时序逻辑电路与组合逻辑电路相比的优势?

触发器及锁存器

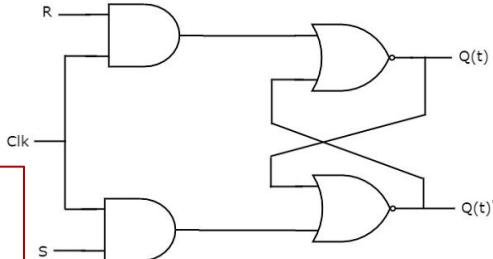


■共同点

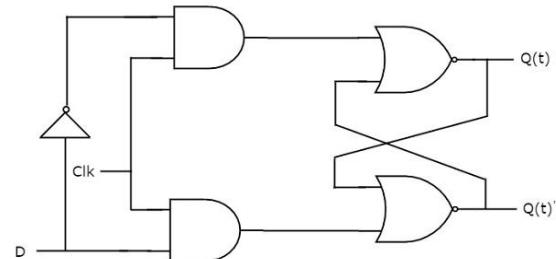
- 具有0和1两个状态，一旦状态确定即可自行保持，带有存储特性

■不同点

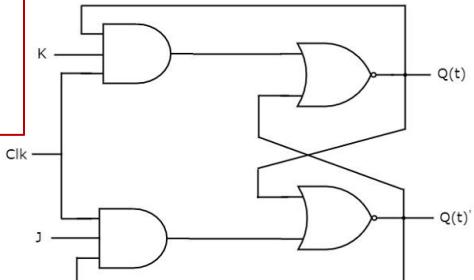
- **锁存器**：对脉冲电平敏感的存储电路，在特定输入脉冲电平作用下改变状态
- **触发器**：对脉冲边沿敏感的存储电路，在特定输入脉冲边沿作用下改变状态



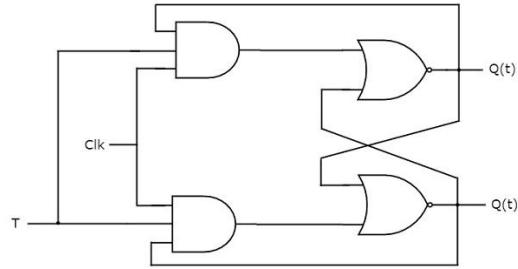
SR触发器



D触发器

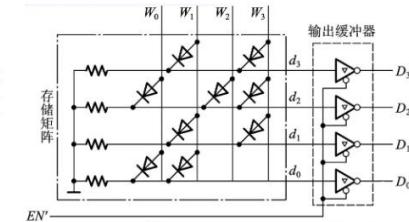
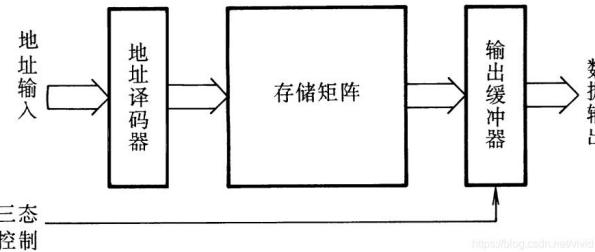


JK触发器



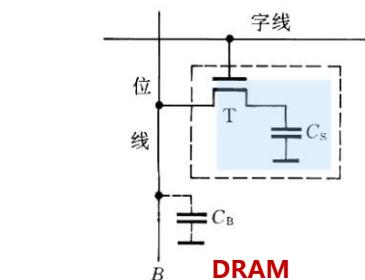
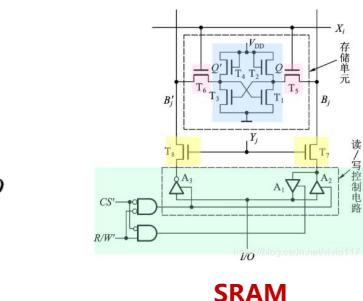
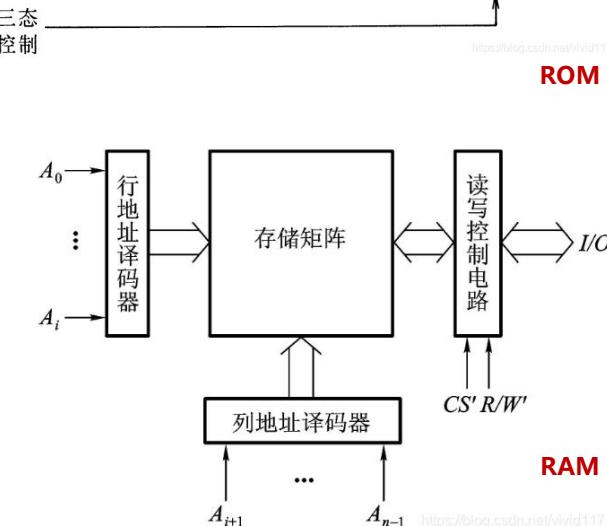
T触发器

存储器



■ 分类

- **ROM**: 只读存储器, 永久性数据存储器, 不能随意改写, PROM, EPROM, EEPROM
- **RAM**: 随机存取存储器, 掉电后数据丢失
- **SRAM**: 静态随机存取存储器
- **DRAM**: 动态随机存取存储器, 需要定期刷新
- **FLASH**: 掉电后数据可保存



数模、模数转换

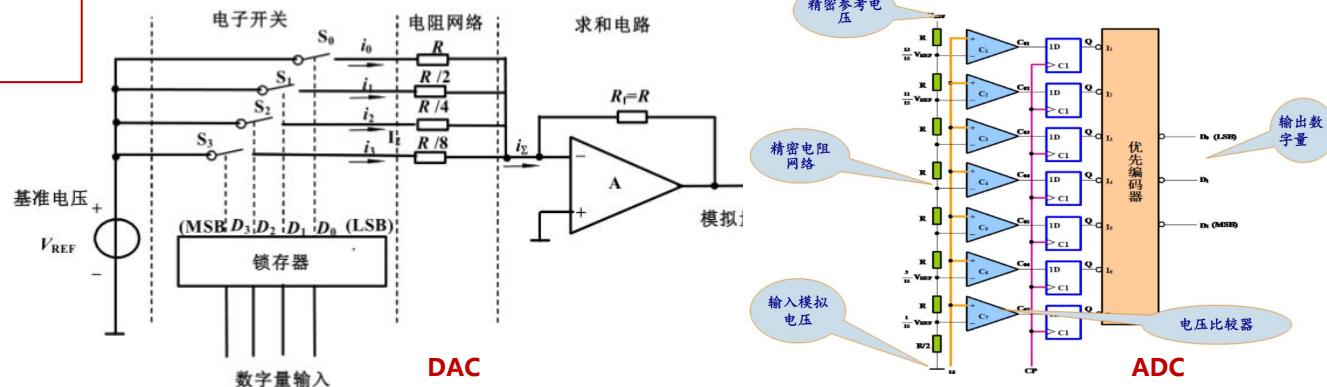
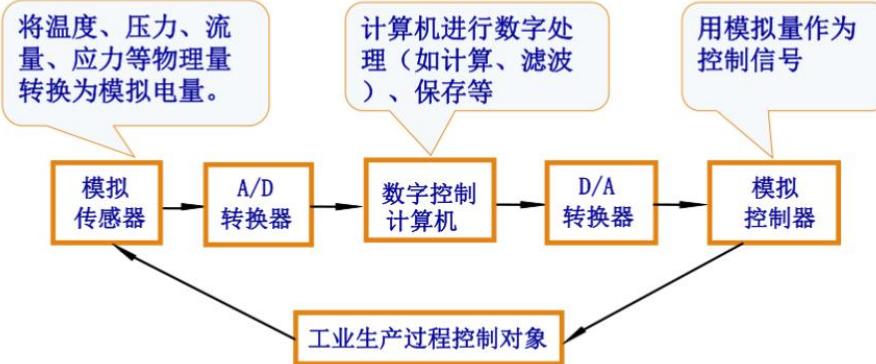


■ 基本概念

- 数模转换: DAC, 将数字量转换为模拟量
- 模数转换: ADC, 将模拟量转换为数字量

■ 技术指标

- 分辨率
- 转换精度
- 线性度
- 电压范围
- 转换时间/速率
- 接口



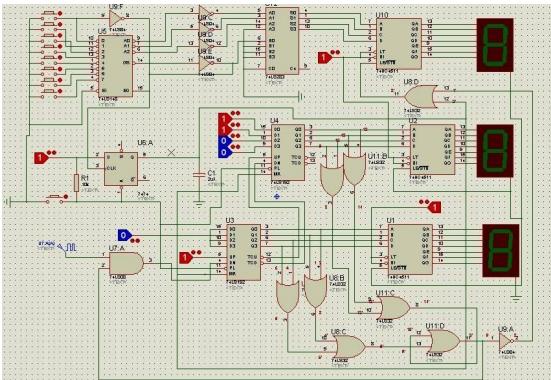
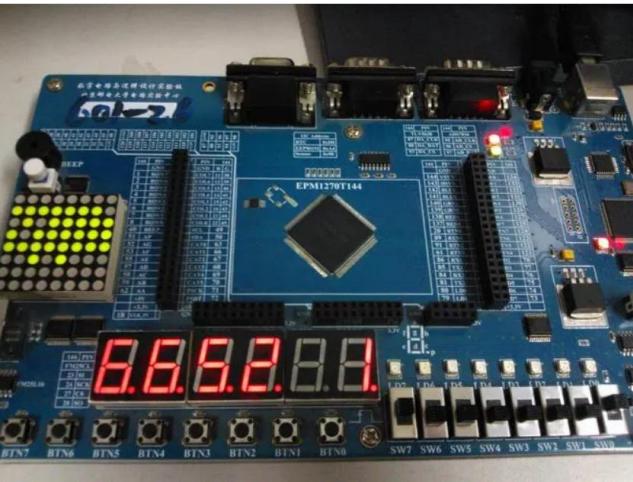
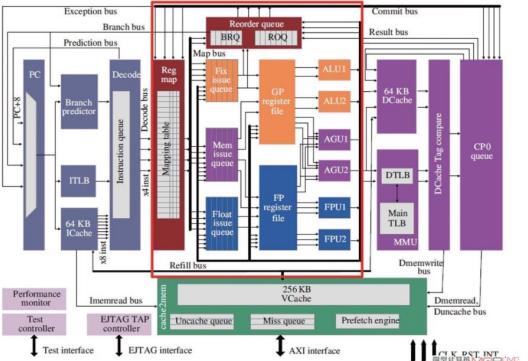
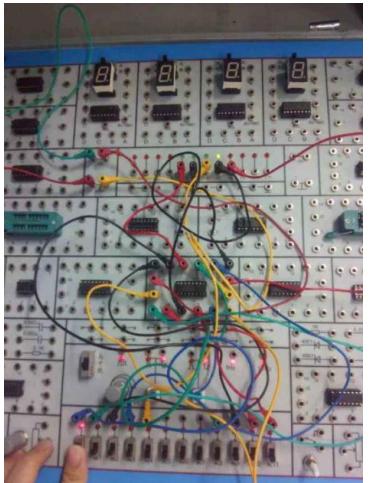
数字系统



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 数字系统

- 交通灯控制系统
- 图形显示系统
- 电子密码锁
- CPU
- 数字控制系统
- 通信系统
- AI
- 数字信号处理系统
-





**FPGA是空白的数字系统
在器件资源范围内可以设计成任意的数字系统**

FPGA与数字系统



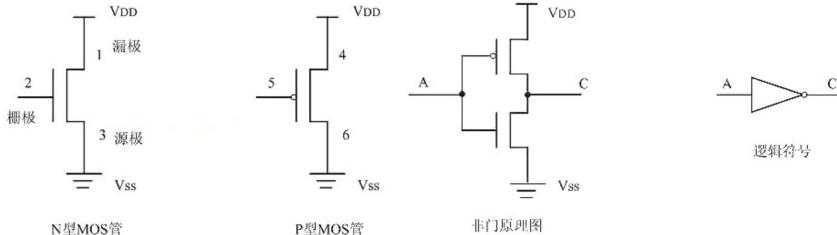
FPGA中逻辑门的硬件实现

◆ 逻辑门电路与晶体管

- ◆ 所有逻辑门电路都可以由PMOS和NMOS实现
- ◆ 非门可以由一个PMOS和一个NMOS实现
- ◆ 其他逻辑门均可由晶体管实现
- ◆ 但是在FPGA中不是使用晶体管实现的

◆ FPGA中的查找表（LUT）

- ◆ 输入作为地址，输出结果为对应地址的内容
- ◆ 右图为与门的LUT的实现
- ◆ FPGA中可使用LUT实现各种逻辑门



输入x	0	1	0	1
输入y	0	0	1	1
地址	0	1	2	3
输出z	0	0	0	1

地址0	0
地址1	0
地址2	0
地址3	1

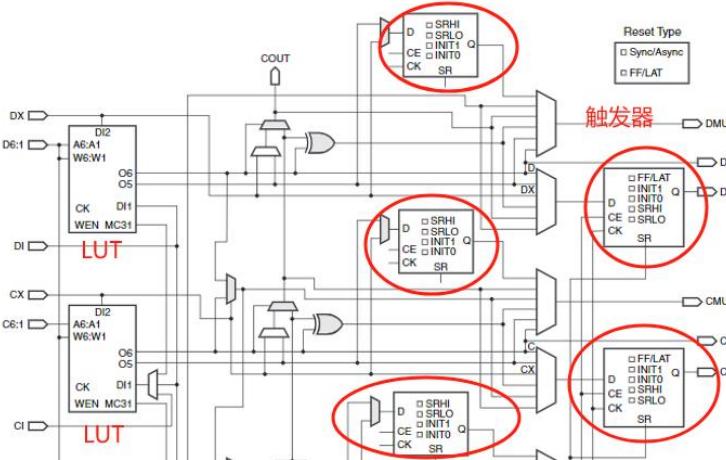
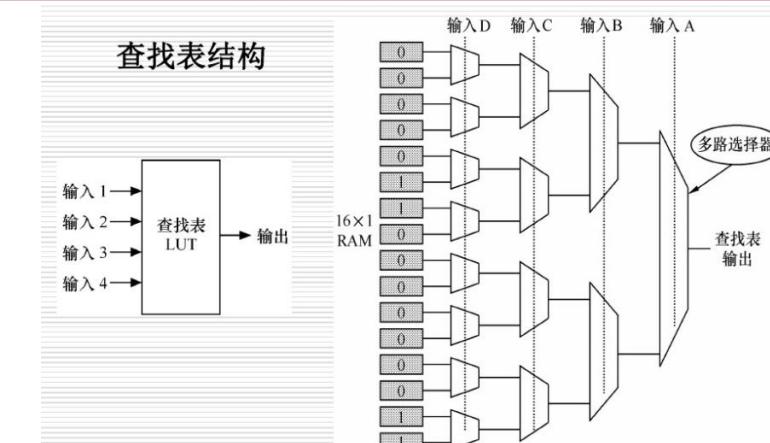
FPGA与数字系统



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

组合逻辑实现方式——LUT

- 查找表（LUT）是构成逻辑单元的重要组成部分
- 本质是一个RAM
- 基于SRAM工艺，也有军品和宇航级FPGA采用Flash或者熔丝与反熔丝工艺
- 开发软件自动计算逻辑电路的所有可能结果，并把结果事先写RAM
- 逻辑运算等于输入一个地址进行查表



时序逻辑实现方式—LUT+器件内置的触发器资源



THANKS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第一讲 (1.3 工具使用介绍&Lab0.0)

2025年3月4日

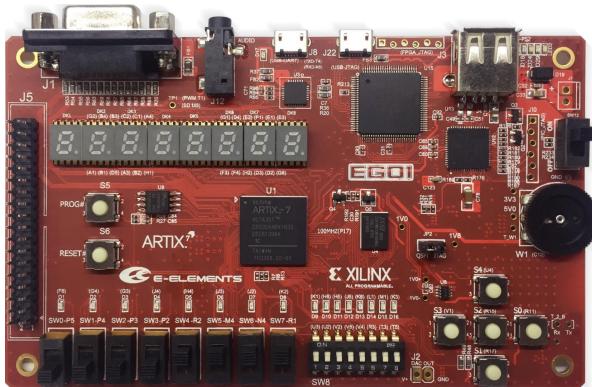
口袋实验室介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 口袋实验室 (EGO1) 特点

- Xilinx 大学计划开发板
- 随身携带，随时操作
- 克服“欺软怕硬”，避免“灌输教学”，避免理论与实践脱节
- 基于Xilinx Artix-7 XC7A35T, 28nm芯片工艺，**主流器件**
- 基于Vivado设计开发工具，**主流工具**
- USB接口供电，内置下载器，**简单易用**
- 板载蓝牙、板载VGA、板载AD/DA、板载音频输出等模块
- 带有32Pin扩展接口，方便功能扩展





- 1. 仔细阅读快速上手指南以及硬件使用手册**
- 2. 注意静电防护**
- 3. 下课可以带走，上课务必带来**
- 4. 注意爱护，课程结束要归还**

口袋实验室介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

EGO1架构

FPGA : Xilinx Artix-7 XC7A35T

时钟 : 100MHz

配置方式 : USB-JTAG/SPI Flash

存储器 :

SRAM : 2Mbit

SPI Flash : N25Q032A

通用IO :

Switch : x8

LED : x16

Button : x5

DIP : x8

通用扩展IO : 32pin

音视频/显示 :

7段数码管 : x8

VGA视频输出接口

Audio音频接口

通信接口 :

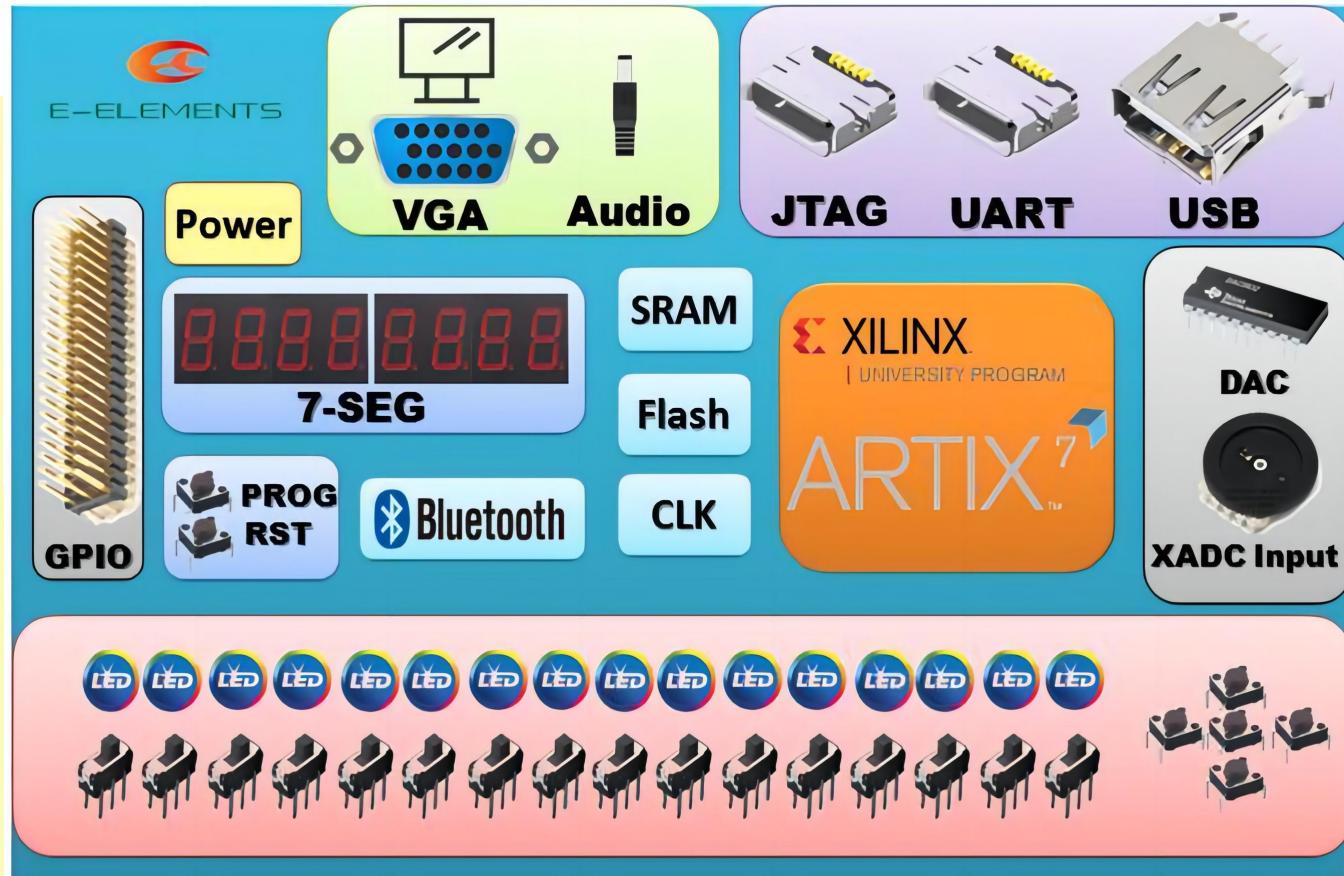
UART : USB转UART

Bluetooth : 蓝牙模块

模拟接口:

DAC: 8-bit分辨率

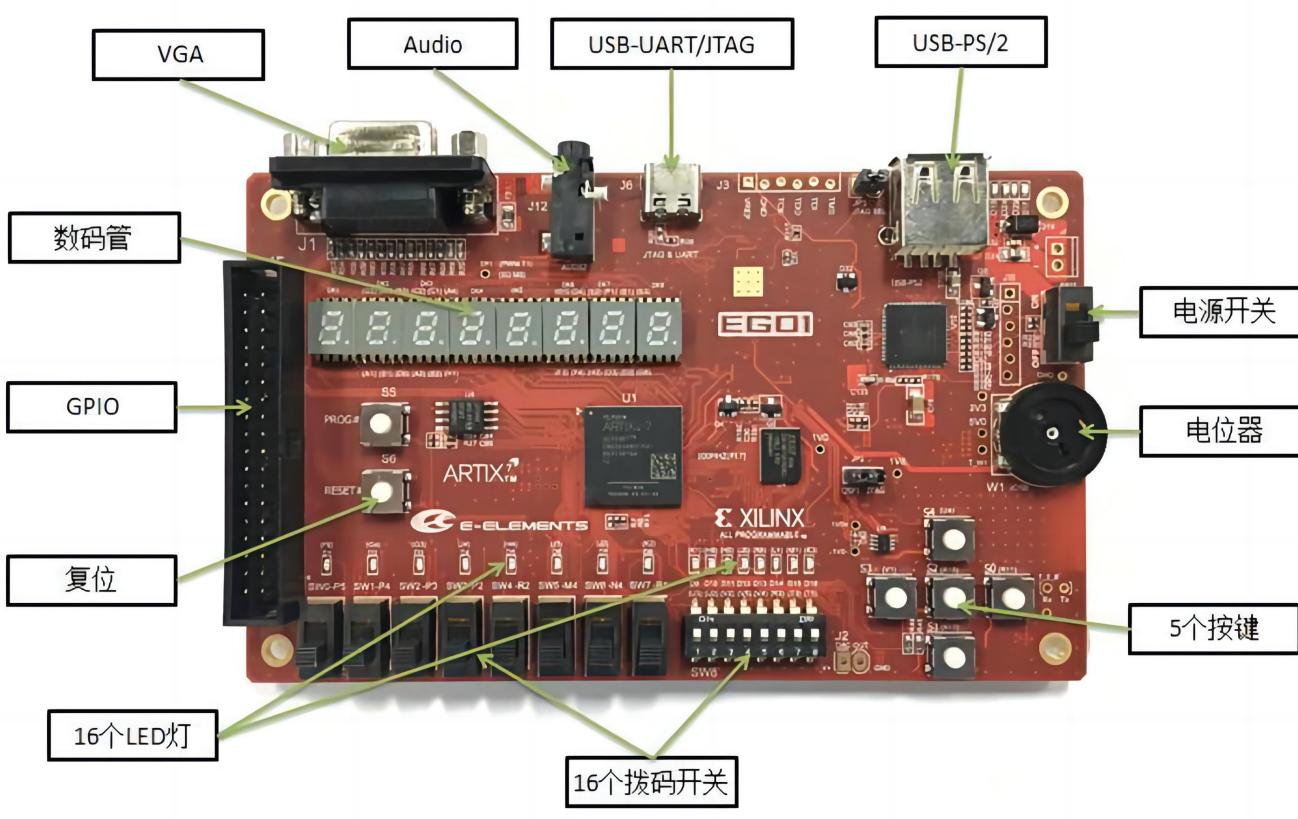
XADC: 2路12bit 1Msps ADC



口袋实验室介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



口袋实验室-硬件组成



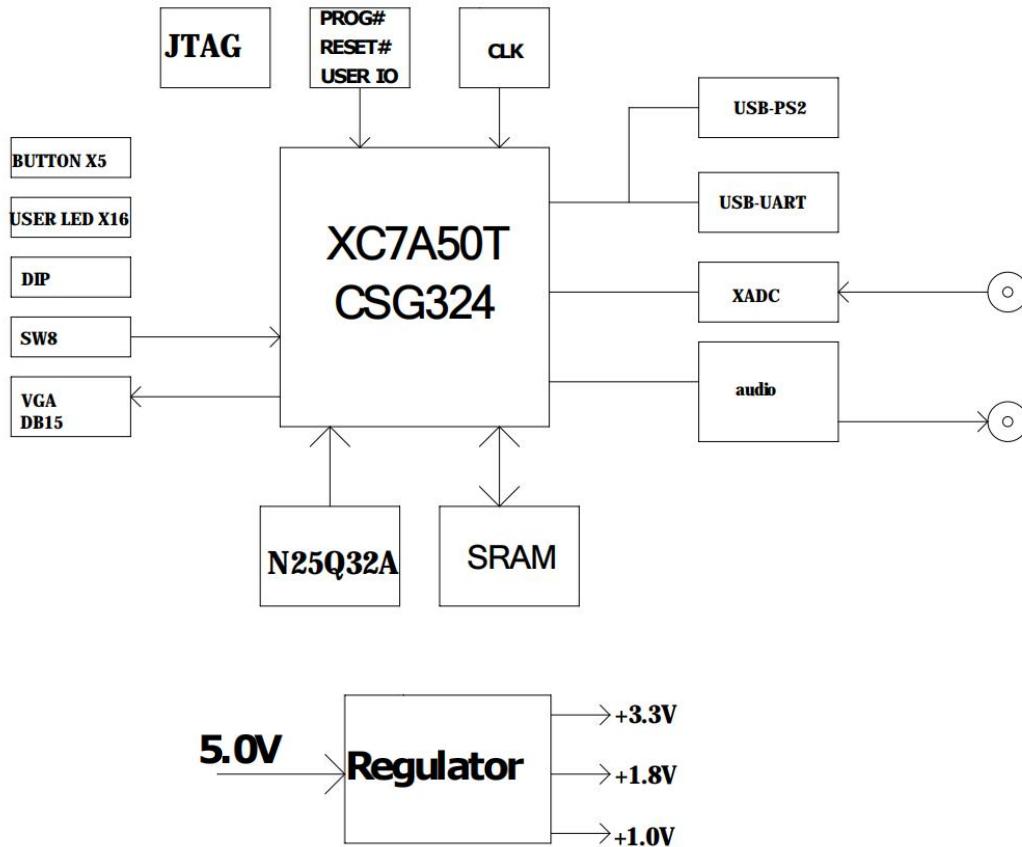
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

■ FPGA硬件能正常工作的必要条件

- FPGA芯片
- 电源
- 配置电路
- 时钟
- 复位电路

■ FPGA硬件其他外围电路

- SDRAM、FLASH等存储器
- 接口 (RS232、网口、SPI、IIC等)
- I/O控制
- 数据采集 (ADC、DAC)
- 人机交互 (显示屏、键盘鼠标)



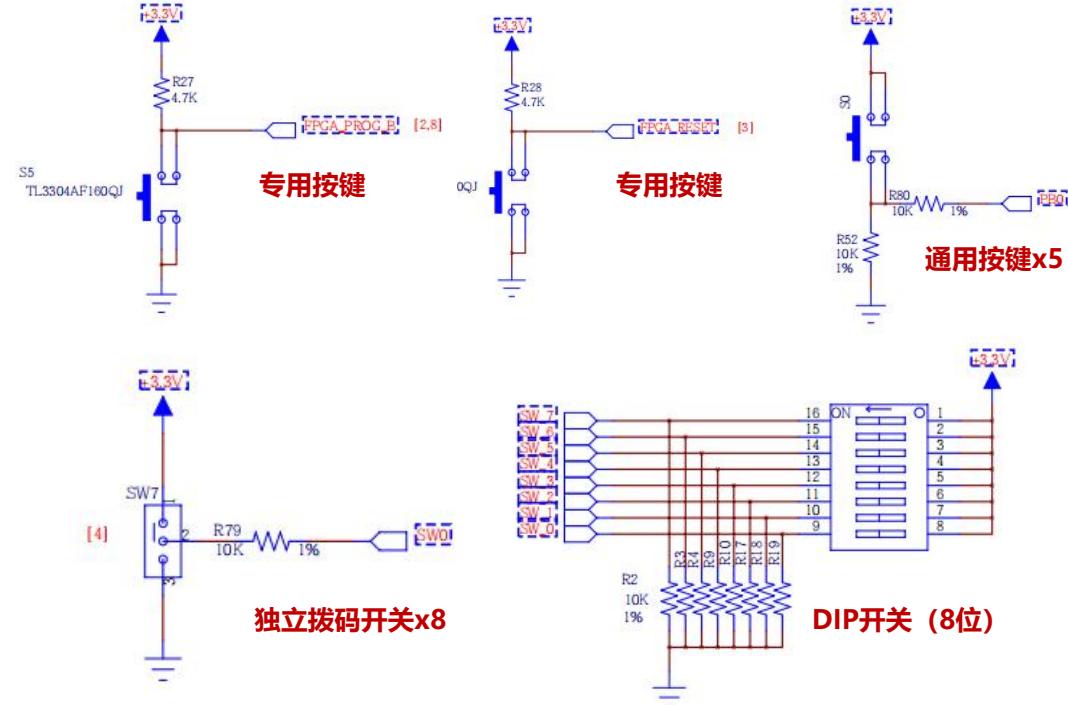
口袋实验室介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 按键

- 按键S5擦除FPGA配置
- 按键S6提供系统外部复位信号，低电平有效
S6->P15
- S0~S4五个通用按键，点触式，按下为高电平
S0->R11,S1->R17,S2->R15,S3->V1,S4->U4
- 8个独立拨码开关，ON->高电平，OFF->低电平
SW0->R1,SW1->N4,SW2->M4,SW3->R2
SW4->P2,SW5->P3,SW6->P4,SW7->P5
- 8位DIP开关，ON->高电平，OFF->低电平
SW0->T5,SW1->T3,SW2->R3,SW3->V4
SW4->V5,SW5->V2,SW6->U2,SW7->U3



EOG1按键资源

口袋实验室介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ LED

- 16个LED

LD1_0->K3, LD1_1->M1, LD1_2->L1, LD1_3->K6

LD1_4->J5, LD1_5->H5, LD1_6->H6, LD1_7->K1

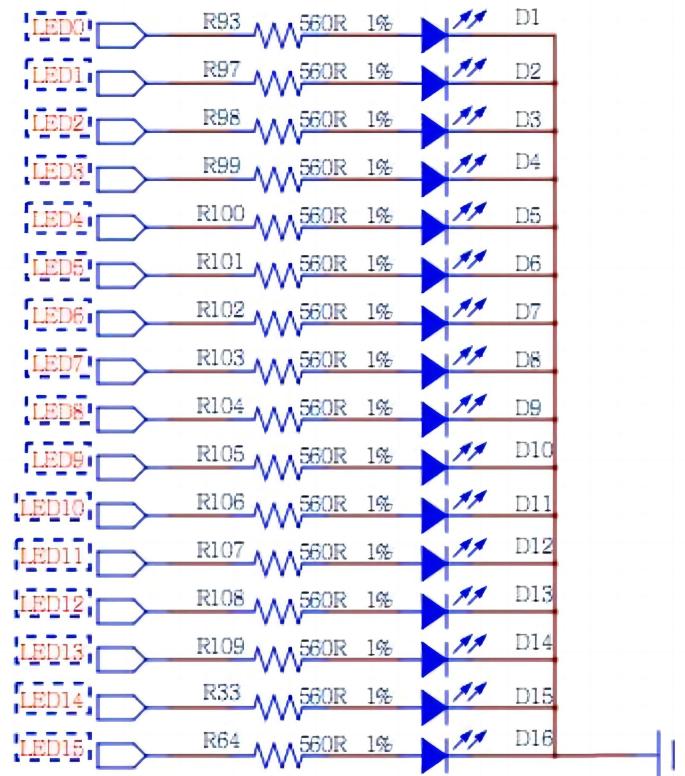
LD2_0->K2, LD2_1->J2, LD2_2->J3, LD2_3->H4

LD2_4->J4, LD2_5->G3, LD2_6->G4, LD2_7->F6

- FPGA管脚输出高电平LED点亮

EOG1 LED资源

LED x 16



软件工具——Vivado



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

概述

实现更快的设计迭代并快速达到您的 FMAX 目标



Vivado 是 AMD 自适应 SoC 和 FPGA 的设计软件。它包括设计入口、综合、布置与路由以及验证 / 仿真工具。

了解 Vivado 设计软件中的高级特性如何在更准确估算 AMD 自适应 SoC 及 FPGA 电源的同时，帮助硬件设计人员缩短编译时间并设计迭代。（资讯图：2023 年 5 月）

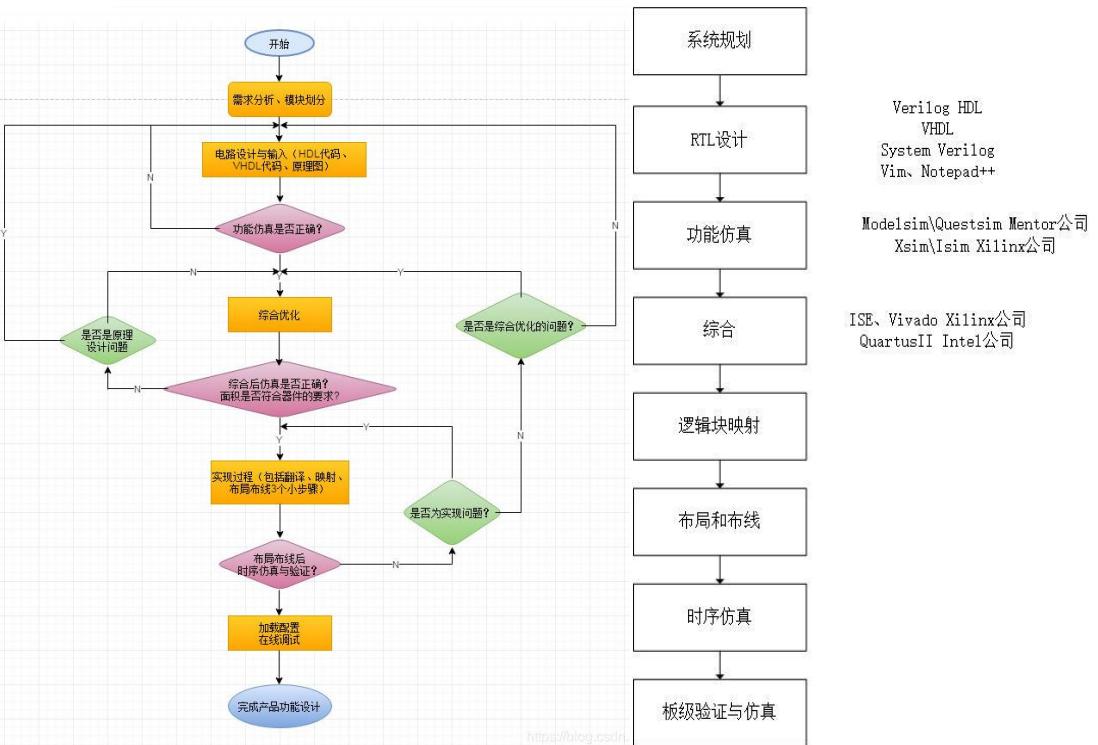
[查看信息图](#)

软件工具——Vivado设计流程



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

现代EDA的开发流程：



使用Vivado软件快速开发流程：

- 通过原理图或编写HDL文件的方式创建设计
- 功能仿真(可选步骤)
- 综合 (类似于电路设计的原理图阶段)
- 通过I/O Planning添加管脚约束或通过编写约束文件添加管脚约束
- 添加时序约束(可选步骤)
- 布局布线(类似于电路设计的PCB设计阶段)
- 时序仿真(可选步骤)
- 生成Bitstream文件
- 将生成的Bitstream文件下载到FPGA芯片

实验0.0：基本门电路的硬件实现

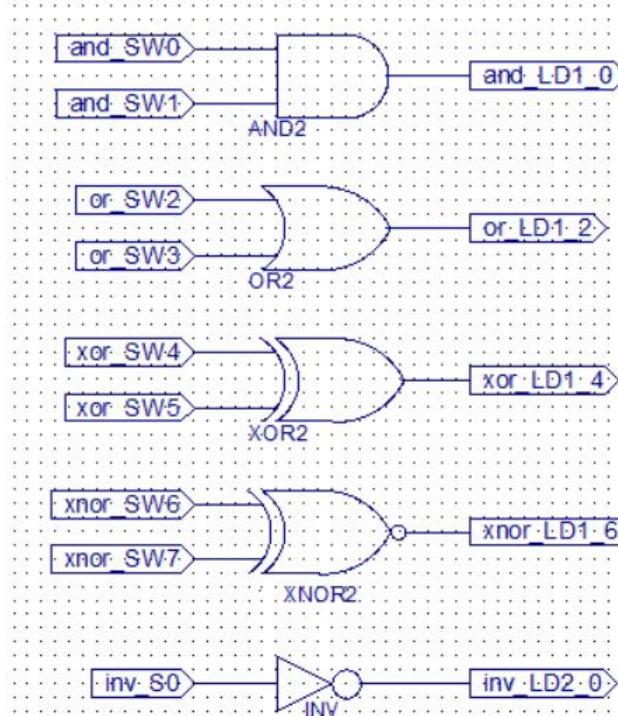


■ 实验内容

- 基本门电路实验，包括与门、或门、异或门、同或门、非门
- 使用DIP8、按键S0作为输入，使用LED灯作为输出
- 基于Vivado原理图输入方式进行设计
- 了解Vivado设计工具的基本使用

■ 电路表达式

- $\text{and_LD1_0} = \text{and_SW0} \& \text{and_SW1}$
- $\text{or_LD1_2} = \text{or_SW2} \mid \text{or_SW3}$
- $\text{xor_LD1_4} = \text{xor_SW4} \wedge \text{xor_SW5}$
- $\text{xnor_LD1_6} = \sim(\text{nor_SW6} \wedge \text{nor_SW7})$
- $\text{inv_LD2_0} = \sim \text{inv_S0}$



电路原理图

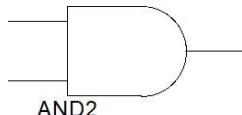
实验0.0：基本门电路的硬件实现



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

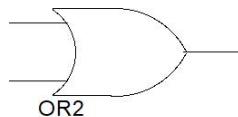
x	0	0	1	1
y	0	1	0	1
z	0	0	0	1

与门



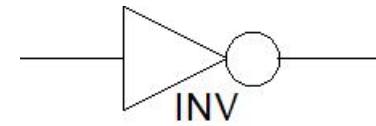
x	0	0	1	1
y	0	1	0	1
z	0	1	1	1

或门



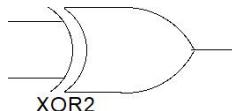
x	0	1
z	1	0

非门



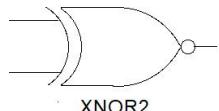
x	0	0	1	1
y	0	1	0	1
z	0	1	1	0

异或门



x	0	0	1	1
y	0	1	0	1
z	1	0	0	1

同或门



实验0.0：基本门电路的硬件实现



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

序号	信号名称	输入/输出	信号描述	FPGA引脚	说明
1	and_SW0	输入	与门输入1	T5	ON为高电平
2	and_SW1	输入	与门输入2	T3	ON为高电平
3	or_SW2	输入	或门输入1	R3	ON为高电平
4	or_SW3	输入	或门输入2	V4	ON为高电平
5	xor_SW4	输入	异或门输入1	V5	ON为高电平
6	xor_SW5	输入	异或门输入2	V2	ON为高电平
7	xnor_SW6	输入	同或门输入1	U2	ON为高电平
8	xnor_SW7	输入	同或门输入2	U3	ON为高电平
9	inv_SO	输入	非门输入	R11	按下为高电平
10	and_LD1_0	输出	与门输出	K3	高电平点亮
11	or_LD1_2	输出	或门输出	L1	高电平点亮
12	xor_LD1_4	输出	异或门输出	J5	高电平点亮
13	xnor_LD1_6	输出	同或门输出	H6	高电平点亮
14	inv_LD2_0	输出	非门输出	K2	高电平点亮

输入输出端口表

实验0.0：基本门电路的硬件实现



The screenshot shows the Vivado 2019.1 software interface. At the top, there is a menu bar with File, Flow, Tools, Window, Help, and a Quick Access search bar. The main window has three main sections:

- Quick Start:** Contains "Create Project >" (highlighted with a red box), "Open Project >", and "Open Example Project >". There is also a red button labeled "建立新工程" (Create New Project).
- Tasks:** Contains "Manage IP >", "Open Hardware Manager >", and "Xilinx Tcl Store >".
- Learning Center:** Contains "Documentation and Tutorials >", "Quick Take Videos >", and "Release Notes Guide >".

At the bottom, there is a toolbar with icons for File, Search, Home, Project, IP, Flow, Tools, Window, Help, and a Tcl Console tab. A status bar at the bottom right shows the time as 18:33 and the date as 2021/3/13.

实验0.0：基本门电路的硬件实现



The screenshot shows the Vivado 2019.1 software interface. On the left, there's a 'Quick Start' panel with options like 'Create Project >', 'Open Project >', and 'Open Example Project >'. Below it is a 'Tasks' panel with 'Manage IP >', 'Open Hardware Manager >', and 'Xilinx Tcl Store >'. At the bottom is a 'Learning Center' panel with 'Documentation and Tutorials >', 'Quick Take Videos >', and 'Release Notes Guide >'. In the center, a 'New Project' dialog box is open with the title 'Create a New Vivado Project'. It contains instructions: 'This wizard will guide you through the creation of a new project. To create a Vivado project you will need to provide a name and a location for your project files. Next, you will specify the type of flow you'll be working with. Finally, you will specify your project sources and choose a default part.' At the bottom of the dialog are buttons: '< Back', 'Next >' (which is highlighted with a red border), 'Finish', and 'Cancel'. The Xilinx logo is visible in the bottom right corner of the dialog. The system tray at the bottom shows various icons and the date/time: 18:35, 2021/3/13.

实验0.0：基本门电路的硬件实现



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

The screenshot shows the Vivado 2019.1 software interface. On the left, there's a 'Quick Start' panel with options like 'Create Project >', 'Open Project >', and 'Open Example Project >'. Below it is a 'Tasks' panel with 'Manage IP >', 'Open Hardware Manager >', and 'Xilinx Tcl Store >'. At the bottom is a 'Learning Center' panel with 'Documentation and Tutorials >', 'Quick Take Videos >', and 'Release Notes Guide >'. A 'Tcl Console' window is at the bottom left. The main area features a 'New Project' dialog box. It has fields for 'Project Name' (containing 'Lab0_sch' and the placeholder '工程名字') and 'Project Location' (containing 'C:/Xilinx/MyProject' and the placeholder '工程目录'). A checked checkbox says 'Create project subdirectory'. A note below says 'Project will be created at: C:/Xilinx/MyProject/Lab0_sch'. At the bottom of the dialog are buttons for '?', '< Back', 'Next >', 'Finish', and 'Cancel'. A large red warning message '路径及名称不要出现中文、空格、符号（下划线除外）！' is overlaid in the center of the dialog. To the right of the dialog, the Xilinx logo is visible.

实验0.0：基本门电路的硬件实现



The screenshot shows the Vivado 2019.1 software interface. On the left, there's a 'Quick Start' panel with options like 'Create Project >', 'Open Project >', and 'Open Example Project >'. Below it is a 'Tasks' panel with 'Manage IP >', 'Open Hardware Manager >', and 'Xilinx Tcl Store >'. At the bottom is a 'Learning Center' panel with 'Documentation and Tutorials >', 'Quick Take Videos >', and 'Release Notes Guide >'. A 'Tcl Console' window is at the bottom left. The main area features a 'New Project' dialog box titled 'Project Type'. It asks 'Specify the type of project to create.' and lists five options:

- RTL Project**
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
 Do not specify sources at this time
- Post-synthesis Project**
You will be able to add sources, view device resources, run design analysis, planning and implementation.
 Do not specify sources at this time
- I/O Planning Project**
Do not specify design sources. You will be able to view part/package resources.
- Imported Project**
Create a Vivado project from a Synplify, XST or ISE Project File.
- Example Project**
Create a new Vivado project from a predefined template.

At the bottom of the dialog are buttons for '?', '< Back', 'Next >', 'Finish', and 'Cancel'. To the right of the dialog, the Xilinx logo is displayed.

实验0.0：基本门电路的硬件实现



The screenshot shows the Vivado 2019.1 software interface. On the left, there's a 'Quick Start' section with 'Create Project >', 'Open Project >', and 'Open Example Project >'. Below it is a 'Tasks' section with 'Manage IP >', 'Open Hardware Manager >', and 'Xilinx Tcl Store >'. At the bottom is a 'Learning Center' section with 'Documentation and Tutorials >', 'Quick Take Videos >', and 'Release Notes Guide >'. A 'Tcl Console' window is at the bottom left.

The main area displays the 'New Project' wizard. It has a title 'Add Sources' and instructions: 'Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.' Below this is a large text input field with a '+' icon for adding files. In the center, the text '添加源文件-暂不添加' is displayed in red. Below the input field are three buttons: 'Add Files', 'Add Directories', and 'Create File'. Underneath these buttons are three checkboxes: 'Scan and add RTL include files into project', 'Copy sources into project', and 'Add sources from subdirectories', with the third one checked. At the bottom are dropdown menus for 'Target language' (set to 'Verilog') and 'Simulator language' (set to 'Mixed'). Navigation buttons include '?', '< Back' (disabled), 'Next >' (highlighted with a red border), 'Finish', and 'Cancel'.

On the right side of the interface, there's a large XILINX logo watermark.

实验0.0：基本门电路的硬件实现



The screenshot shows the Vivado 2019.1 software interface. On the left, there's a 'Quick Start' section with links to 'Create Project', 'Open Project', and 'Open Example Project'. Below it is a 'Tasks' section with links to 'Manage IP', 'Open Hardware Manager', and 'Xilinx Tcl Store'. At the bottom left is a 'Learning Center' section with links to 'Documentation and Tutorials', 'Quick Take Videos', and 'Release Notes Guide'. A 'Tcl Console' window is also visible at the bottom left.

In the center, a 'New Project' dialog box is open. It has a title 'Add Constraints (optional)' and a sub-instruction 'Specify or create constraint files for physical and timing constraints.' Below this is a large text area with the placeholder '添加约束文件-暂不添加' (Add constraint file - temporarily not added). At the bottom of the dialog are buttons for 'Add Files', 'Create File', and 'Copy constraints files into project'. The 'Next >' button is highlighted with a red border. At the very bottom of the dialog are buttons for '?', '< Back', 'Next >', 'Finish', and 'Cancel'.

On the right side of the interface, there's a large XILINX logo watermark.

实验0.0：基本门电路的硬件实现



Vivado 2019.1

File Flow Tools Window Help Q: Quick Access

VIVADO HLx Editions

Quick Start

Create Project >
Open Project >
Open Example Project >

Tasks

Manage IP >
Open Hardware Manager >
Xilinx Tcl Store >

Learning Center

Documentation and Tutorials >
Quick Take Videos >
Release Notes Guide >

New Project

Default Part

Choose a default Xilinx part or board for your project.

选择FPGA器件

Parts | Boards

Reset All Filters

Category: All Package: csg324 Temperature: All Remaining

Family: Artix-7 Speed: -1 Static power: All Remaining

Search: Q-

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Tran
xc7a15tcsg324-1	324	210	10400	20800	25	0	45	0
xc7a35tcsg324-1	324	210	20800	41600	50	0	90	0
xc7a50tcsg324-1	324	210	32600	65200	75	0	120	0
xc7a75tcsg324-1	324	210	47200	94400	105	0	180	0
xc7a100tcsg324-1	324	210	63400	126800	135	0	240	0

< Back Next > Finish Cancel

Tcl Console

工程向导会引导你完成设计源文件和目标器件的选择

Windows Search Task View Start File Explorer Twitter Word Excel

18:44 2021/3/13

XILINX

实验0.0：基本门电路的硬件实现



The screenshot shows the Vivado 2019.1 software interface. On the left, there's a 'Quick Start' section with 'Create Project >', 'Open Project >', and 'Open Example Project >'. Below it is a 'Tasks' section with 'Manage IP >', 'Open Hardware Manager >', and 'Xilinx Tcl Store >'. At the bottom, there's a 'Learning Center' section with 'Documentation and Tutorials >', 'Quick Take Videos >', and 'Release Notes Guide >'. A small 'Tcl Console' window is visible at the bottom left.

New Project

New Project Summary

- A new RTL project named 'Lab0_sch' will be created.
- No source files or directories will be added. Use Add Sources to add them later.
- No constraints files will be added. Use Add Sources to add them later.

The default part and product family for the new project:
Default Part: xc7a35tcs324-1
Product: Artix-7
Family: Artix-7
Package: csg324
Speed Grade: -1

建立新工程总结

To create the project, click Finish

Finish

Other UI elements include 'File', 'Flow', 'Tools', 'Window', 'Help', 'Q- Quick Access' in the top menu bar, and system icons like search, task view, and network in the taskbar at the bottom right.

实验0.0：基本门电路的硬件实现



加载EGo1 IP元件库

The screenshot shows the Xilinx Vivado IDE interface. The 'Tools' menu is open, and the 'Custom Commands' option is highlighted with a red box. Below it, the 'EGo1' command is also highlighted with a red box. The main workspace displays a 'Project Summary' for 'Lab0_sch' with various project details. At the bottom, a 'Design Runs' table shows two entries: 'synth_1' and 'impl_1', both in the 'Not started' status.

Project Summary

Settings Edit

Project name: Lab0_sch
Project location: C:/Xilinx/MyProject/Lab0_sch
Product family: Artix-7
Project part: xc7a35tcsg324-1
Top module name: Not defined
Target language: Verilog
Simulator language: Mixed

Synthesis Implementation

Status: Not started Status: Not started
Messages: No errors or warnings Messages: No errors or warnings

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synthesis 2019)
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado Implementation 2019)

PROGRAM AND DEBUG 配置自定义Tcl命令

Windows Search Task View Start Twitter LinkedIn 18:53 2021/3/13 3

实验0.0：基本门电路的硬件实现



The screenshot shows the Vivado 2019.1 software interface. On the left is the Project Navigator with sections for Project Manager, IP Integrator (highlighted in blue), Simulation, RTL Analysis, Synthesis, Implementation, and Program and Debug. The central area is the 'BLOCK DESIGN - project' window, which includes tabs for Sources, Design (selected), and Signals. The Design tab contains a 'Diagram' pane with a red border labeled '电路原理图编辑区' (Circuit Schematic Editor Area). Below the diagram is a message: 'This design is empty. Press the + button to add IP.' At the bottom of the Design tab is a 'Properties' panel. The bottom of the window features a 'Tcl Console' tab showing command-line output and a 'Type a Tcl command here' input field. The Windows taskbar at the bottom shows various pinned icons.

Ready

Default Layout

File Edit Flow Tools Reports Window Layout View Help Q Quick Access

BLOCK DESIGN - project

Sources Design Signals

project

Properties

Select an object to see properties

Diagram

This design is empty. Press the + button to add IP.

Tcl Console

INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1700] Loaded user IP repository 'c:/EGoI_LIB/Library'.
Wrote : <C:\Xilinx\MyProject\Lab0.sch\Lab0.sch.srcs\sources_1\bd\project\project.bd>

ERROR: This script was generated using Vivado <2017.4> and is being run in <2019.1> of Vivado. Please run the script in Vivado <2017.4> then open the design in Vivado <2019.1>. Upgrade the de:
1
update_compile_order -fileset sources_1

Type a Tcl command here

Windows Taskbar icons: File Explorer, Edge, File Explorer, Twitter, Word, Snipping Tool, Task View, Taskbar settings, Network, Power, Volume, Language, Date/Time (18:55, 2021/3/13)

实验0.0：基本门电路的硬件实现



重新命名设计文件

The screenshot shows the Vivado Block Design interface. The left sidebar has a red box around the 'Save Block Design As...' option under the 'File' menu. The central workspace shows a 'BLOCK DESIGN - project' window with tabs for Sources, Design, and Signals. Under Sources, there is a tree view of 'Design Sources (1)' containing 'project (project.bd)'. Below it are 'Constraints' and 'Simulation Sources (1)' with 'sim_1 (1)'. A 'Diagram' tab is visible but empty. A 'Source File Properties' panel shows 'project.bd' with 'Enabled' checked and a location path. The bottom right shows a 'Tcl Console' tab with command history and a text input field.

INFO: [IP_Flow 19-1700] Loaded user IP repository 'c:/EGoI_LIB/Library'.
Wrote : <C:\Xilinx\MyProject\Lab0_sch\Lab0_sch.srcs\sources_1\bd\project\project.bd>

ERROR: This script was generated using Vivado <2017.4> and is being run in <2019.1> of Vivado. Please run the script in Vivado <2017.4> then open the design in Vivado <2019.1>. Upgrade the de
@ 1
update_compile_order -fileset sources_1
open_bd_design [C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srcs/sources_1/bd/project/project.bd]

Type a Tcl command here

21:26
2021/3/13

实验0.0：基本门电路的硬件实现



國科大杭州高等研究院

Hangzhou Institute for Advanced Study, UCAS

The screenshot shows the Xilinx Vivado IDE interface with the following details:

- File Menu:** File, Edit, Flow, Tools, Reports, Window, Layout, View, Help.
- Quick Access Bar:** Contains icons for Project Manager, IP Catalog, and Diagram.
- Flow Navigator:** Shows the current project is "BLOCK DESIGN - project".
- Project Manager:** Settings, Add Sources, Language Templates, IP Catalog.
- IP Integrator:** Create Block Design, Open Block Design, Generate Block Design.
- Simulation:** Run Simulation.
- RTL Analysis:** Open Elaborated Design.
- Synthesis:** Run Synthesis, Open Synthesized Design.
- Implementation:** Run Implementation, Open Implemented Design.
- Tcl Console:** Displays the following log output:

```
INFO: [IP_Flow 19-1700] Loaded user IP repository 'c:/EG01_LIB/Library'.
Wrote : <C:\Xilinx\MyProject\Lab0_sch\Lab0_sch.srcs\sources_1\bd\project.bd>

ERROR: This script was generated using Vivado <2017.4> and is being run in <2019.1> of Vivado. Please run the script in Vivado <2017.4> then open the design in Vivado <2019.1>. Upgrade the de...
1
update_compile_order -fileset sources_1
open_bd_design {C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srcs/sources_1/bd/project/project.bd}
```
- Diagram View:** Shows a tree structure for "Design Sources (1)" including "project (project.bd)", "Constraints", "Simulation Sources (1)", and "Utility Sources".
- Save Block Design As Dialog:** A modal dialog titled "Save Block Design As" is open, prompting to save the "project" block design to a new name and location. The "Design name:" field contains "Lab0_sch", the "Directory:" dropdown shows "<Local to Project>", and the "OK" button is highlighted with a red box.
- Red Text Overlay:** The text "重新命名设计文件" (Rename design file) is displayed in red at the top right of the dialog.
- System Tray:** Shows icons for network, battery, and system status.
- Bottom Status Bar:** Shows "将该设计另存为新设计" (Save design as new), the date "2021/3/13", and the time "19:02".

实验0.0：基本门电路的硬件实现



The screenshot shows the Xilinx Vivado interface. On the left, the Project Manager sidebar is open, showing sections like PROJECT MANAGER, IP INTEGRATOR (selected), and SIMULATION. In the center, the BLOCK DESIGN window displays a 'Sources' tab with a tree view of project files: 'Design Sources (2)' containing 'Lab0_sch (Lab0_sch.bd)' and 'project (project.bd)', and 'Simulation Sources (2)'. A 'Diagram' tab is also visible. A 'Remove Sources' dialog box is centered over the sources list, asking 'OK to remove the one selected file from the project?' with an option to 'Also delete the project local file/directory from disk'. Below the dialog, the 'Tcl Console' tab shows command-line logs related to the project's build process. The status bar at the bottom right indicates the time as 19:05 and the date as 2021/3/13.

记得删除默认的文件

从工程中删除当前的选择

File Edit Flow Tools Reports Window Layout View Help Quick Access

BLOCK DESIGN project Lab0_sch

PROJECT MANAGER

- Settings
- Add Sources
- Language Templates
- IP Catalog

IP INTEGRATOR

- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION

- Run Simulation

RTL ANALYSIS

- Open Elaborated Design

SYNTHESIS

- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION

- Run Implementation
- Open Implemented Design

Default Layout

Remove Sources

Source File Properties

project.bd

Type: Block Designs

Part: xc7a35tcsg324-1

General Properties

OK Cancel

This design is empty. Press the + button to add IP.

Tcl Console

```
Wrote : <C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srcs/sources_1/bd/Lab0_sch/Lab0_sch.bd>
Wrote : <C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srcs/sources_1/bd/Lab0_sch/Lab0_sch.bd>
Wrote : <<C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srcs/sources_1/bd/Lab0_sch/ui/bd_2684423c.ui>
add_files -norecurse C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srcs/sources_1/bd/Lab0_sch/Lab0_sch.bd
export_ip_user_files -of_objects [get_files C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srcs/sources_1/bd/Lab0_sch/Lab0_sch.bd]
open_bd_design [C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srcs/sources_1/bd/Lab0_sch/Lab0_sch.bd]
Successfully read diagram <Lab0_sch> from BD file <C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srcs/sources_1/bd/Lab0_sch/Lab0_sch.bd>
```

Type a Tcl command here

PROBLEMS AND DEBUG

19:05 2021/3/13

实验0.0：基本门电路的硬件实现



The screenshot shows the Xilinx Vivado Design Suite interface. A floating window titled "选择浮动窗口" (Select Floating Window) is overlaid on the main workspace. The main workspace displays the "BLOCK DESIGN - Lab0_sch" project. The "Sources" tab is selected, showing a list of design sources including "Lab0_sch (Lab0_sch.bd)". The "Diagram" tab is also visible. The "Tcl Console" tab at the bottom shows the following Tcl command history:

```
set_property location [2 274 27] [get_bd_cells xup_xor2_0]
delete_bd_objs [get_bd_cells xup_and2_0]
delete_bd_objs [get_bd_cells xup_or2_0]
delete_bd_objs [get_bd_cells xup_inv_0]
delete_bd_objs [get_bd_cells xup_xor2_0]
open_bd_design C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srcs/sources_1/bd/Lab0_sch/Lab0_sch.bd
open_bd_design C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srcs/sources_1/bd/Lab0_sch/Lab0_sch.bd
```

The Windows taskbar at the bottom shows various application icons, and the system tray indicates the date and time as 19:23, 2021/3/13.

实验0.0：基本门电路的硬件实现



Diagram

Search: Q: XUP (80 matches)

- uart_v1_0
- vga_v1_0
- XUP 1-input INV
- XUP 2-input AND
- XUP 2-input NAND
- XUP 2-input NOR
- XUP 2-input OR
- XUP 2-input XNOR
- XUP 2-input XOR
- XUP 2_to_1_mux
- XUP 3-input AND
- XUP 3-input NAND

ENTER to select, ESC to cancel, Ctrl+Q for IP details

xup_and2_0

xup_or2_0

xup_xor2_0

xup_xnor2_0

xup_inv_0

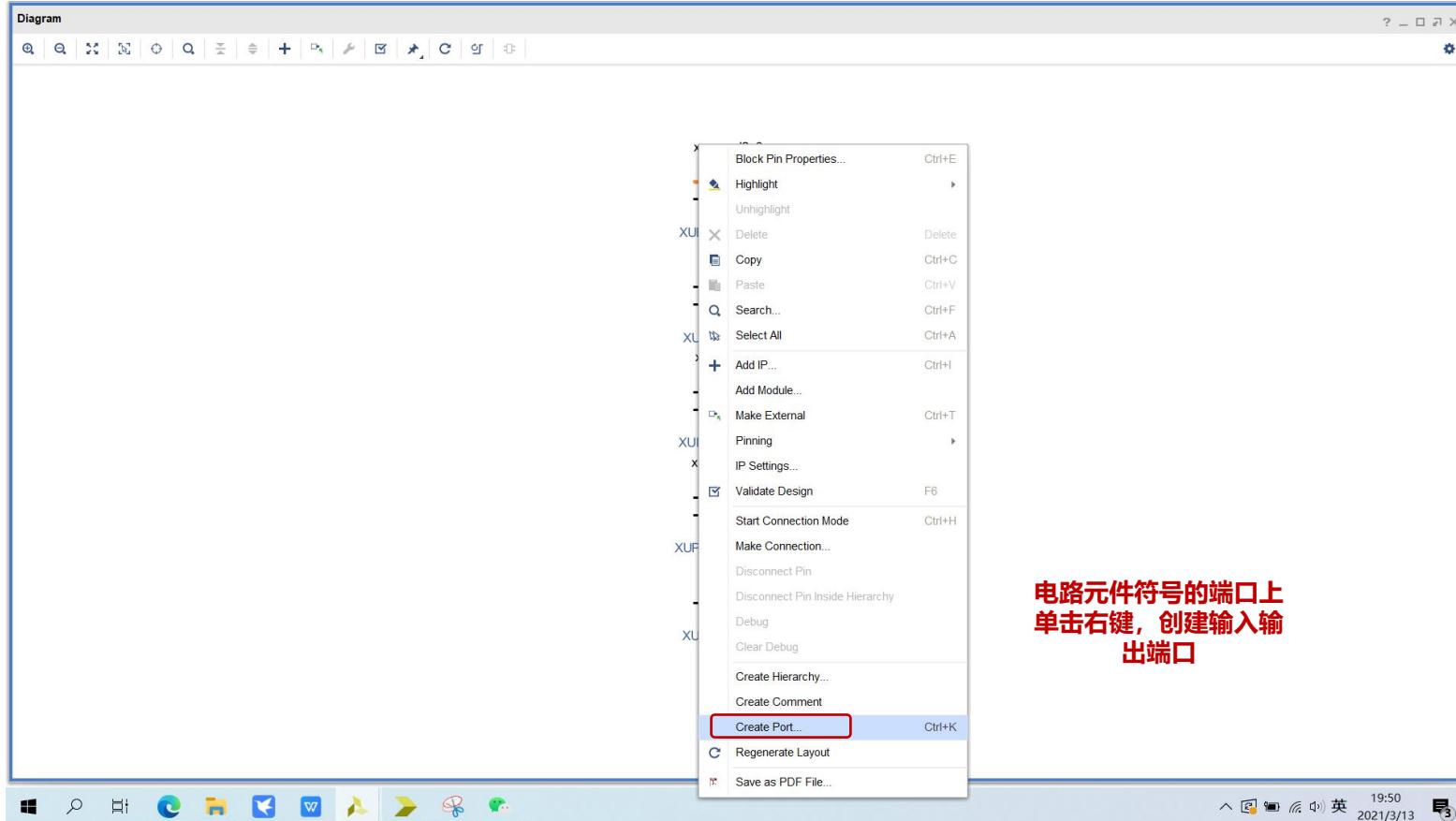
放置电路元件符号

The screenshot shows a schematic editor interface. On the left, a search results list is displayed under the heading 'Search: Q: XUP (80 matches)'. The list includes various IP components such as 'uart_v1_0', 'vga_v1_0', and several logic gates like 'XUP 1-input INV', 'XUP 2-input AND', etc. Some items in the list are highlighted with red boxes. On the right, a vertical stack of logic gate symbols is shown, each labeled with its name: 'xup_and2_0', 'xup_or2_0', 'xup_xor2_0', 'xup_xnor2_0', and 'xup_inv_0'. Each symbol is a small rectangle with two input pins labeled 'a' and 'b' and one output pin labeled 'y'. The text '放置电路元件符号' (Place circuit component symbols) is overlaid in red on the right side of the screen.

实验0.0：基本门电路的硬件实现



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

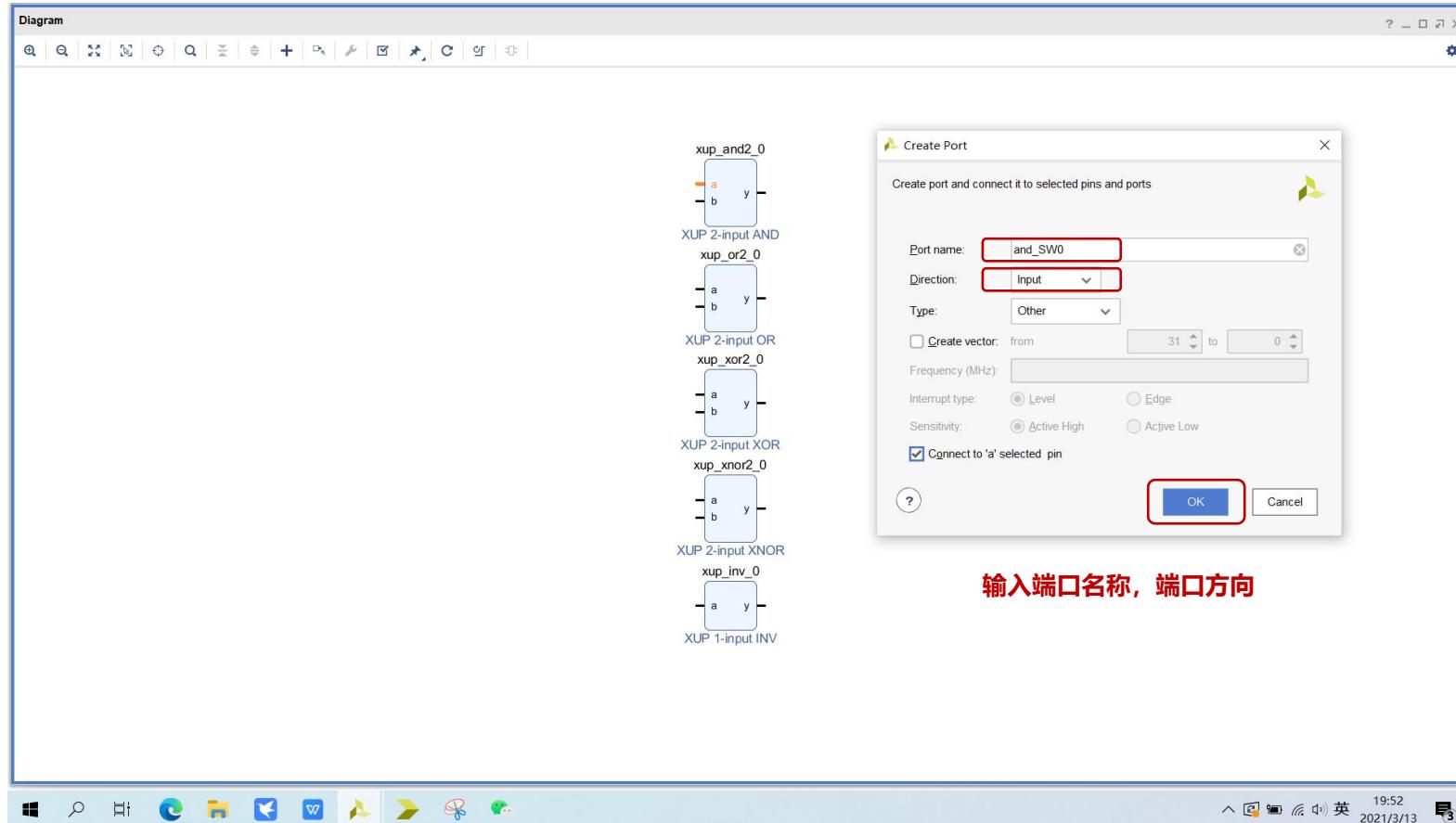


实验0.0：基本门电路的硬件实现



國科大杭州高等研究院

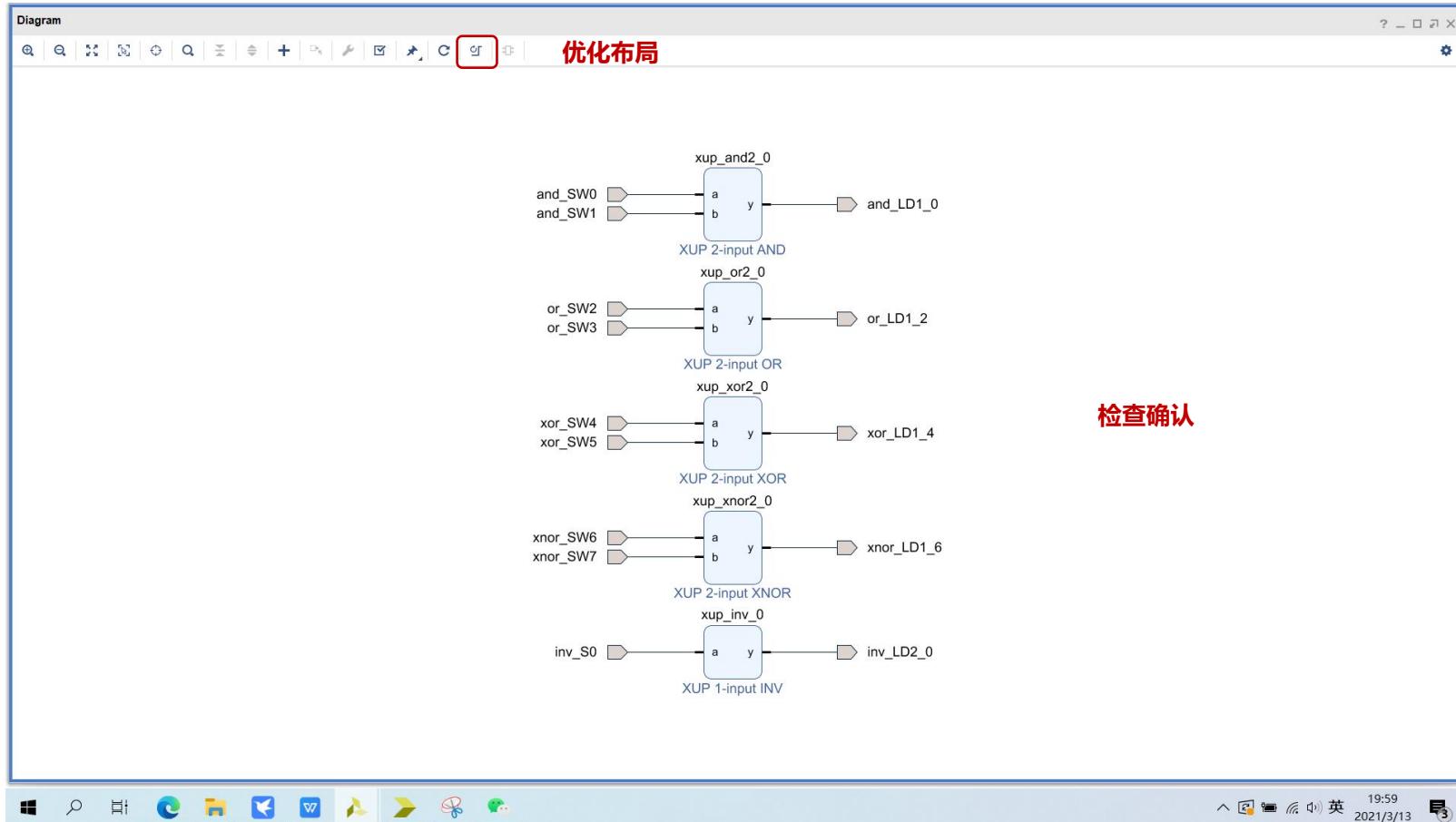
Hangzhou Institute for Advanced Study, UCAS



实验0.0：基本门电路的硬件实现



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



实验0.0：基本门电路的硬件实现



创建HDL封装

The screenshot shows the Xilinx Vivado software interface. On the left, the 'PROJECT MANAGER' and 'IP INTEGRATOR' sections are visible. In the center, a logic diagram is displayed with two AND gates and two OR gates. The top AND gate, labeled 'xup_and2_0', has inputs 'and_SW0' and 'and_SW1' and output 'y'. The bottom OR gate, labeled 'xup_or2_0', has inputs 'or_SW2' and 'or_SW3' and output 'y'. The logic diagram consists of three main components: 'XUP 2-input AND' and 'XUP 2-input OR'. A context menu is open over the project manager area, with the 'Create HDL Wrapper...' option highlighted.

File Edit Flow Tools Reports Window Layout View Help Quick Access

BLOCK DESIGN - Lab0_sch *

PROJECT MANAGER

- Settings
- Add Sources
- Language Templates
- IP Catalog

IP INTEGRATOR

- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION

- Run Simulation

RTL ANALYSIS

- Open Elaborated Design

SYNTHESIS

- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION

- Run Implementation
- Open Implemented Design

PROGRAM AND DEBUG

- 生成HDL向导文件并复制到项目中

Source Node Properties... Ctrl+E

Open File Alt+O

Create HDL Wrapper... **Selected**

View Instantiation Template

Generate Output Products...

Reset Output Products...

Replace File...

Copy File Into Project

Copy All Files Into Project Alt+I

Remove File from Project... Delete

Enable File Alt+=号

Disable File Alt+-号

Hierarchy Update

Refresh Hierarchy

IP Hierarchy

Set as Top

Add Module to Block Design

Set File Type...

Set Used In...

Edit Constraints Sets...

Edit Simulation Sets...

Associate ELF Files...

Add Sources... Alt+A

Report IP Status

Diagram

xup_and2_0

a
b
y

XUP 2-input AND

xup_or2_0

a
b
y

XUP 2-input OR

and_SW0
and_SW1
y

or_SW2
or_SW3
y

20:00
2021/3/13

实验0.0：基本门电路的硬件实现



File Edit Flow Tools Reports Window Layout View Help Quick Access Ready Default Layout

BLOCK DESIGN - Lab0_sch *

PROJECT MANAGER

- Settings
- Add Sources
- Language Templates
- IP Catalog

IP INTEGRATOR

- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION

- Run Simulation

RTL ANALYSIS

- Open Elaborated Design

SYNTHESIS

- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION

- Run Implementation
- Open Implemented Design

PROGRAM AND DEBUG

生成HDL白盒文件并复制在项目里

Sources Design Signals ? □ □

Design Sources (1)
Lab0_sch (Lab0_sch.bd)

Diagram

Create HDL Wrapper

You can either add or copy the HDL wrapper file to the project. Use copy option if you would like to modify this file.

Options

Copy generated wrapper to allow user edits
 Let Vivado manage wrapper and auto-update

OK Cancel

Source File Properties

Lab0_sch.bd

Type: Block Designs
Part: xc7a35tcsg324-1

General Properties

Tcl Console

```
endgroup
startgroup
create_bd_port -dir 0 inv_LD2_0
connect_bd_net [get_bd_pins /xup_inv_0/y] [get_bd_ports inv_LD2_0]
endgroup
regenerate_bd_layout -routing
regenerate_bd_layout -routing
```

Type a Tcl command here

Diagram

xup_and2_0
and_SW0
and_SW1
XUP 2-input AND
xup_or2_0
or_SW2
or_SW3
XI IP 2-inout OR

使用Vivado默认设置

Ready Default Layout

20:02 2021/3/13

实验0.0：基本门电路的硬件实现



國科大杭州高等研究院

Hangzhou Institute for Advanced Study, UCAS

The screenshot displays the Xilinx Vivado 2021.3 software interface, specifically the Block Design environment. The left sidebar contains several sections:

- PROJECT MANAGER**: Settings, Add Sources, Language Templates, IP Catalog.
- IP INTEGRATOR**: Create Block Design, Open Block Design, Generate Block Design.
- SIMULATION**: Run Simulation.
- RTL ANALYSIS**: Open Elaborated Design.
- SYNTHESIS**: Run Synthesis, Open Synthesized Design.
- IMPLEMENTATION**: Run Implementation, Open Implemented Design.

The central workspace is divided into three main tabs:

- Sources**: Shows a tree view of design sources. A node labeled "Lab0_sch_wrapper (Lab0_sch_wrapper.v) (1)" is selected, revealing its internal structure: "Lab0_sch_i : Lab0_sch (Lab0_sch.bd) (1)" and "Lab0_sch (Lab0_sch.v) (5)".
- Diagram**: Displays a logic diagram for a two-input AND-OR gate. It consists of two XUP 2-input AND blocks (labeled "xup_and2_0") and two XUP 2-input OR blocks (labeled "xup_or2_0"). The inputs are labeled "and_SW0" and "and_SW1" for the AND blocks, and "or_SW2" and "or_SW3" for the OR blocks. The outputs are labeled "y".
- Tcl Console**: Shows the command-line interface used for synthesis. The terminal window displays the following commands and their outputs:

```
make_wrapper -files [get_files C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srs/sources_1/bd/Lab0_sch/Lab0_sch.bd] -top
Wrote : <C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srs/sources_1/bd/Lab0_sch/Lab0_sch.bd>
Wrote : <C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srs/sources_1/bd/Lab0_sch/ui/ui_top.bd>
VHDL Output written to : C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srs/sources_1/bd/Lab0_sch/synth/Lab0_sch.v
VHDL Output written to : C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srs/sources_1/bd/Lab0_sch/sim/Lab0_sch.v
VHDL Output written to : C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srs/sources_1/bd/Lab0_sch/hdl/Lab0_sch_wrapper.v
add_files -norecurse C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.srs/sources_1/bd/Lab0_sch/hdl/Lab0_sch_wrapper.v
```

At the bottom, there is a search bar with the placeholder "Type a Tcl command here".

实验0.0：基本门电路的硬件实现



1. 对设计文件进行综合

2. 启动综合

The screenshot shows the Xilinx Vivado IDE interface. On the left, the Flow Navigator pane is open, showing sections for IP INTEGRATOR, SIMULATION, RTL ANALYSIS, SYNTHESIS, IMPLEMENTATION, PROGRAM AND DEBUG, and a summary of the current project status. The SYNTHESIS section is currently selected, and the 'Run Synthesis' button is highlighted with a red box. In the center, the 'BLOCK DESIGN - Lab0_sch' window is displayed, showing a hierarchical tree of design sources and constraints. A red circle highlights the 'Launch Runs' dialog box, which is overlaid on the main window. This dialog box contains fields for 'Source File Properties' (Type: Verilog, Library: xl_default.lib), 'Launch directory' (Default Launch Directory), and 'Options' (Launch runs on local host: Number of jobs: 4). At the bottom of the dialog box, there is an 'OK' button highlighted with a red box and a 'Cancel' button. Below the dialog box, the 'Tcl Console' and 'Messages' tabs are visible, showing command-line output related to the synthesis process. The right side of the interface shows a schematic diagram of a logic circuit with two AND gates and two OR gates, labeled 'xup_and2_0' and 'xup_or2_0'. The circuit has inputs 'and_SW0', 'and_SW1', 'or_SW2', and 'or_SW3', and outputs 'y'.

实验0.0：基本门电路的硬件实现



查看工程状态

The screenshot shows the Vivado IDE interface with the following details:

- Project Summary:** Lab0_sch_wrapper
- Target language:** Verilog
- Simulator language:** Verilog
- Synthesis:**
 - Status: Queued
 - Messages: 12 warnings
 - Active run: synth_1
 - Part: xc7a35tcsg324-1
 - Strategy: Vivado Synthesis Defaults
 - Report Strategy: Vivado Synthesis Default Reports
 - Incremental synthesis: None
- Implementation:**
 - Status: Not started
 - Messages: No errors or warnings
 - Active run: impl_1
 - Part: xc7a35tcsg324-1
 - Strategy: Vivado Implementation Defaults
 - Report Strategy: Vivado Implementation Default Reports
 - Incremental implementation: None
- Tcl Console:** Shows Vivado Commands and Block Design logs.
- Messages:** Warning (12), Info (91), Status (43).

实验0.0：基本门电路的硬件实现



Synthesis Complete ✓

Default Layout

BLOCK DESIGN - Lab0_sch

Diagram

Synthesis Completed

Synthesis successfully completed.

Next

Run Implementation (radio button selected)

Open Synthesized Design

View Reports

Don't show this dialog again

OK Cancel

xup_and2_0 (XUP 2-input AND:1.0)
xup_inv_0 (XUP 1-input INV:1.0)

xup_and2_0
V0
V1
a
b
y
and_LD1_0

XUP 2-input AND

xup_inv_0
S0
a
y
inv_LD2_0

XUP 1-input INV

选择取消
需要先进行管脚约束再进行布局布线

Tcl Console

```
Adding component instance block -- xilinx.com:xup:xup_inv:1.0 - xup_inv_0
Successfully read diagram 'Lab0_sch' from BD file <C:/Xilinx/MyProject/2024/Lab0_sch/Lab0_sch.srs/sources_1/bd/Lab0_sch/Lab0_sch.bd>
reset_run synth_1
launch_runs synth_1 -jobs 6
[Mon Mar 4 22:22:14 2024] Launched synth_1...
Run output will be captured here: C:/Xilinx/MyProject/2024/Lab0_sch/Lab0_sch.runs/synth_1/runme.log
```

实验0.0：基本门电路的硬件实现



Synthesis Complete ✓

Default Layout

Flow Navigator BLOCK DESIGN - Lab0_sch

PROJECT MANAGER

- Settings
- Add Sources
- Language Templates
- IP Catalog

IP INTEGRATOR

- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION

- Run Simulation

RTL ANALYSIS

- Open Elaborated Design

SYNTHESIS

- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION

- Run Implementation
- Open Implemented Design

PROGRAM AND DEBUG

- Generate Bitstream
- Open Hardware Manager

Sources Design Signals

Diagram

xup_and2_0 (XUP 2-input AND:1.0)

xup_inv_0 (XUP 1-input INV:1.0)

and_SW0 → a, b → y → and_LD1_0

inv_S0 → a → y → inv_LD2_0

打开综合后的设计

Tcl Console

```
Adding component instance block -- xilinx.com:xup:xup_inv:1.0 - xup_inv_0
Successfulely read diagram <Lab0_sch> from BD file <C:/Xilinx/MyProject/2024/Lab0_sch/Lab0_sch.srs/sources_1/bd/Lab0_sch/Lab0_sch.bd>
: reset_run synth_1
: launch_runs synth_1 -jobs 6
: [Mon Mar 4 22:22:14 2024] Launched synth_1...
: Run output will be captured here: C:/Xilinx/MyProject/2024/Lab0_sch/Lab0_sch.runs/synth_1/runme.log
```

实验0.0：基本门电路的硬件实现



打开I/O planning界面
对管脚进行约束

The screenshot shows a software interface for hardware design. On the left, there is a navigation pane with sections like PROJECT MANAGER, IP INTEGRATOR, SIMULATION, RTL ANALYSIS, and SYNTHESIS. Under SYNTHESIS, 'Open Synthesized Design' is selected. In the center, a sub-menu for 'SYNTHESIZED' is open, with 'I/O Planning' highlighted and surrounded by a red box. Below this, there are options for Floorplanning, Debug, and Timing Analysis. A note says 'Drop I/O banks on voltages or the "NONE" folder to set/unset Internal VREF.' To the right, there is a large grid representing a chip layout with various colored pads (blue, green, red, grey) and labels like 'C', 'S', and 'V'. At the bottom, there is a table titled 'Tcl Console' with columns for Name, Direction, Neg Diff Pair, Package Pin, Fixed, Bank, I/O Std, Vcco, Vref, Drive Strength, Slew Type, Pull Type, Off-Chip Termination, and IN_TERM. The table shows a single entry under 'All ports (5)' for 'Scalar ports (5)'.

实验0.0：基本门电路的硬件实现



1. 打开综合后的原理图

2. 输入管脚号

3. 更改电平标准为LVCMS33

The screenshot shows the Vivado 2019.1 interface with the following details:

- Sources:** Lab0_sch_wrapper.v (Verilog)
- Design Sources:** Lab0_sch_wrapper.v, Lab0_sch.bd
- Source File Properties:** Enabled, Location: C:/Xilinx/MyProject/2024/Lab0_sch/Lab0_sch.srccs/s, Type: Verilog
- I/O Ports:** A table showing I/O port details:

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination	IN_TERM
OUT						del -	1.800		12	SLOW	NONE	FP_VTT_50	
IN						del -	1.800				NONE	NONE	
IN						del -	1.800				NONE	NONE	
OUT						del -	1.800		12	SLOW	NONE	FP_VTT_50	
IN						del -	1.800				NONE	NONE	

实验0.0：基本门电路的硬件实现



The screenshot shows the Xilinx Vivado IDE interface. On the left, the Flow Navigator pane is open, displaying the project structure under the RTL ANALYSIS section. The Schematic editor is open in the center, showing a logic circuit with components like IBUF, AND, INV, OR, and a Lab0_sch_i block. Below it, the I/O Ports editor is open, showing a table of I/O pins and their properties. Two columns of the table are highlighted with red boxes: 'Package Pin' and 'I/O Std'. A red box also highlights the column headers 'Name', 'Direction', 'Neg Diff Pair', 'Package Pin', 'Fixed', 'Bank', and 'I/O Std'. Red text annotations '输入管脚号' (Input Pin Number) and '更改电平标准' (Change Level Standard) are overlaid on the table.

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination	IN_TERM	
OUT			K3	✓	34	LVCMS33*	-	3.300	12	▼	SLOW	NONE	FP_VTT_50	▼
IN			T5	✓	34	LVCMS33*	-	3.300				NONE	NONE	▼
IN			T3	✓	34	LVCMS33*	-	3.300				NONE	NONE	▼
OUT			K2	✓	35	LVCMS33*	-	3.300	12	▼	SLOW	NONE	FP_VTT_50	▼
IN			R11	✓	14	LVCMS33*	-	3.300				NONE	NONE	▼
OUT			L1	✓	34	LVCMS33*	-	3.300	12	▼	SLOW	NONE	FP_VTT_50	▼
IN			R3	✓	34	LVCMS33*	-	3.300				NONE	NONE	▼
IN			V4	✓	34	LVCMS33*	-	3.300				NONE	NONE	▼
OUT			H6	✓	35	LVCMS33*	-	3.300	12	▼	SLOW	NONE	FP_VTT_50	▼
IN			U2	✓	34	LVCMS33*	-	3.300				NONE	NONE	▼
IN			U3	✓	34	LVCMS33*	-	3.300				NONE	NONE	▼
OUT			J5	✓	35	LVCMS33*	-	3.300	12	▼	SLOW	NONE	FP_VTT_50	▼
IN			V5	✓	34	LVCMS33*	-	3.300				NONE	NONE	▼
IN			V2	✓	34	LVCMS33*	-	3.300				NONE	NONE	▼

实验0.0：基本门电路的硬件实现



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

保存管脚约束文件

ELABORATED DESIGN * - xc7a35tcsg324-1

PROJECT MANAGER

- Settings
- Add Sources
- Language Templates
- IP Catalog

IP INTEGRATOR

- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION

- Run Simulation

RTL ANALYSIS

- Open Elaborated Design
- Report Methodology
- Report DRC
- Report Noise
- Schematic

SYNTHESIS

- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION

- Run Implementation

I/O Port Properties

Name: and_SW1

Direction: IN

General Properties Configure

Schematic

Tcl Console

Messages Log Reports Design Runs Package Pins I/O Ports

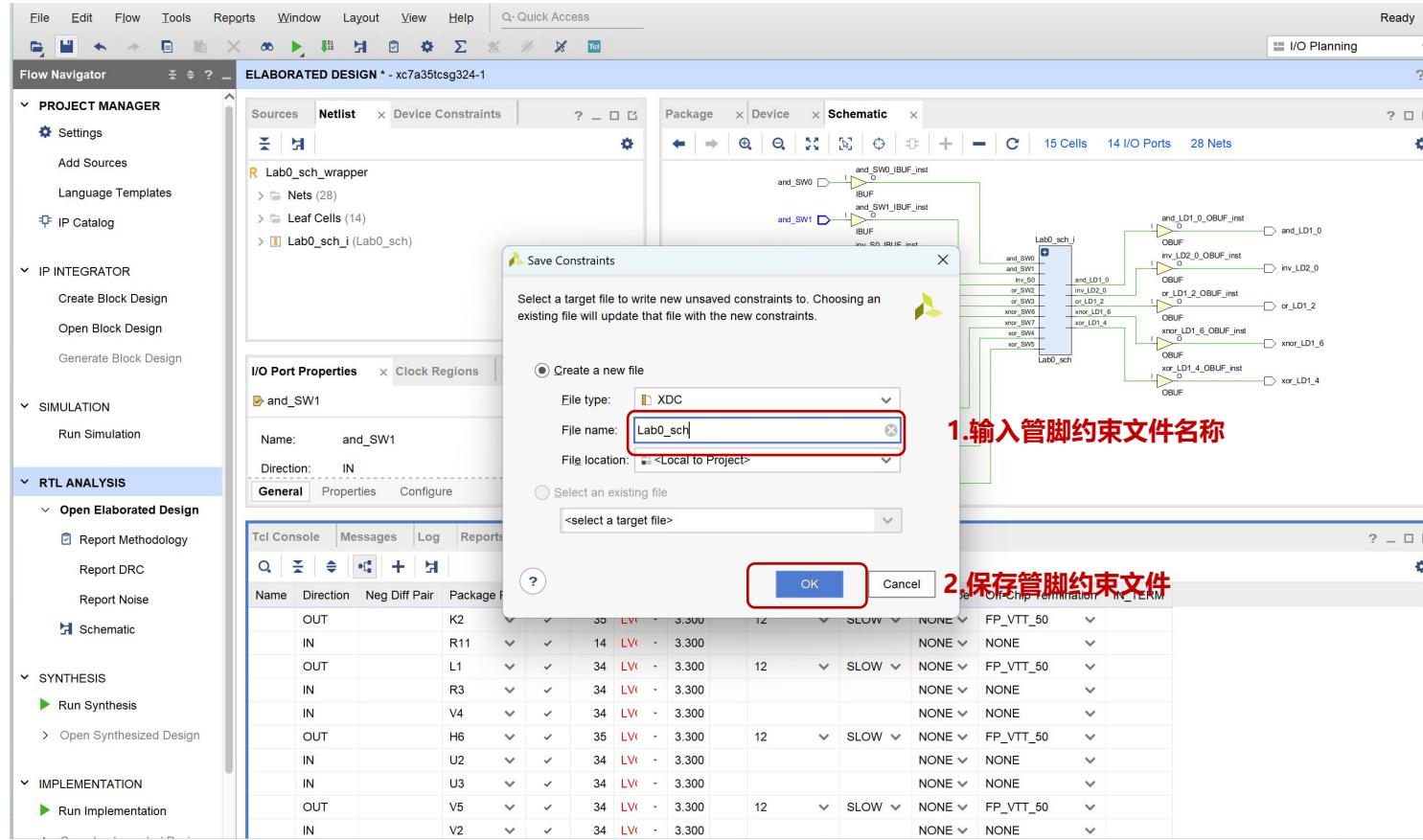
Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination	IN_TERM
OUT			K2	✓	✓	35 LV	-	3.300	12	✓	SLOW	NONE	FP_VTT_50
IN			R11	✓	✓	14 LV	-	3.300				NONE	NONE
OUT			L1	✓	✓	34 LV	-	3.300	12	✓	SLOW	NONE	FP_VTT_50
IN			R3	✓	✓	34 LV	-	3.300				NONE	NONE
IN			V4	✓	✓	34 LV	-	3.300				NONE	NONE
OUT			H6	✓	✓	35 LV	-	3.300	12	✓	SLOW	NONE	FP_VTT_50
IN			U2	✓	✓	34 LV	-	3.300				NONE	NONE
IN			U3	✓	✓	34 LV	-	3.300				NONE	NONE
OUT			V5	✓	✓	34 LV	-	3.300	12	✓	SLOW	NONE	FP_VTT_50
IN			V2	✓	✓	34 LV	-	3.300				NONE	NONE

实验0.0：基本门电路的硬件实现



國科大杭州高等研究院

Hangzhou Institute for Advanced Study, UCAS



实验0.0：基本门电路的硬件实现



Ready I/O Planning

ELABORATED DESIGN - xc7a35tcsg324-1

Lab0_sch_wrapper

I/O Port Properties and Clock Regions

and_SW1

Name: and_SW1
Direction: IN

General Properties Configure

Schematic View

Tcl Console Messages Log Reports Design Runs Package Pins I/O Ports

实现 (布局布线)

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination	IN_TERM		
OUT			K2	✓		35	LVI	-	3.300	12	✓	SLOW	NONE	FP_VTT_50	✓
IN			R11	✓		14	LVI	-	3.300			NONE	NONE	NONE	✓
or_LD1_2			L1	✓		34	LVI	-	3.300	12	✓	SLOW	NONE	FP_VTT_50	✓
IN			R3	✓		34	LVI	-	3.300			NONE	NONE	NONE	✓
OUT			V4	✓		34	LVI	-	3.300			NONE	NONE	NONE	✓
IN			H6	✓		35	LVI	-	3.300	12	✓	SLOW	NONE	FP_VTT_50	✓
IN			U2	✓		34	LVI	-	3.300			NONE	NONE	NONE	✓
OUT			U3	✓		34	LVI	-	3.300			NONE	NONE	NONE	✓
IN			V5	✓		34	LVI	-	3.300	12	✓	SLOW	NONE	FP_VTT_50	✓
IN			V2	✓		34	LVI	-	3.300			NONE	NONE	NONE	✓

实验0.0：基本门电路的硬件实现



Lab0_sch - [C:/Xilinx/MyProject/2024/Lab0_sch/Lab0_sch.xpr] - Vivado 2019.1

File Edit Flow Tools Reports Window Layout View Help Quick Access

Flow Navigator Run Simulation

RTL ANALYSIS Open Elaborated Design

SYNTHESIS Run Synthesis Open Synthesized Design Constraints Wizard Edit Timing Constraints Set Up Debug Report Timing Summary Report Clock Networks Report Clock Interaction Report Methodology Report DRC Report Noise Report Utilization Report Power Schematic

IMPLEMENTATION Run Implementation Open Implemented Design

PROGRAM AND DEBUG Generate Bitstream

SYNTHESIZED DESIGN - synth_1 | xc7a35tcsg324-1

Sources Netlist Device Constraints Internal VREF 0.6V 0.675V

I/O Port Properties Clock Regions inv_S0 Name: inv_S0 Direction: IN Package pin: R11 Fixed

and_SW0 and_SW1 and_SW0 and_SW1 and_SW0 and_SW1 and_LD1_0 inv_LD2_0 and_LD1_0 inv_LD2_0

IBUF IBUF IBUF IBUF OBUF OBUF OBUF OBUF

and_SW0_IBUF_inst and_SW1_IBUF_inst and_LD1_0_OBUF_inst inv_LD2_0_OBUF_inst

and_SW0 and_SW1 inv_S0 Lab0_sch_i Lab0_sch_j and_LD1_0 and_LD2_0

Synthesis is out-of-date. OK to launch synthesis first? Implementation will automatically start when synthesis completes.

Yes No Cancel

重新综合

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination	IN_TERM
OUT			K3	✓	34	LVC MOS33*	-	3.300	12	✓	SLOW	✓	NONE ✓ FP_VTT_50 ✓
IN			T5	✓	34	LVC MOS33*	-	3.300				NONE ✓	NONE ✓
IN			T3	✓	34	LVC MOS33*	-	3.300				NONE ✓	NONE ✓
OUT			K2	✓	35	LVC MOS33*	-	3.300	12	✓	SLOW	✓	NONE ✓ FP_VTT_50 ✓
IN			R11	✓	14	LVC MOS33*	-	3.300				NONE ✓	NONE ✓

实验0.0：基本门电路的硬件实现



國科大杭州高等研究院

Hangzhou Institute for Advanced Study, UCAS

The screenshot shows the Vivado 2019.1 interface with the following details:

- Project:** Lab0_sch - [C:/Xilinx/MyProject/2024/Lab0_sch/Lab0_sch.xpr] - Vivado 2019.1
- Flow Navigator:** Shows "SYNTHESIS" as the active flow.
- Synthesis Status:** "Synthesized Design is out-of-date. Synthesis results were reset. Close Design"
- Schematic View:** Displays the circuit diagram with components like Lab0_sch_i, and_SW0, and_SW1, inv_S0, and LD1_0, LD1_2, LD2_0, LD2_2.
- I/O Port Properties Dialog:** A modal dialog is open for the "inv_S0" port, showing options for launching synthesis runs. The "OK" button is highlighted with a red box.
- Table View:** Shows I/O port properties for all ports, including Name, Direction, Pin, and various synthesis parameters like Slew Type, Pull Type, and Off-Chip Termination.

实验0.0：基本门电路的硬件实现



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Implementation Complete ✓

Default Layout

Flow Navigator Add Sources

Language Templates

IP Catalog

IP INTEGRATOR Create Block Design

Open Block Design

Generate Block Design

SIMULATION Run Simulation

RTL ANALYSIS Open Elaborated Design

SYNTHESIS Run Synthesis

Open Synthesized Design

IMPLEMENTATION Run Implementation

Open Implemented Design

PROGRAM AND DEBUG Generate Bitstream

Open Hardware Manager

BLOCK DESIGN - Lab00_sch

Sources Design Signals

Design Sources (1)

> Lab00_sch_wrapper (Lab00_sch_wrapper.v) (1)

Constraints

Simulation Sources (1)

> sim_1 (1)

Hierarchy IP Sources Libraries Compile Order

Source File Properties

Lab00_sch.bd

Enabled

Location: C:/Xilinx/MyProject2023/Lab00_sch/Lab00_sch.s

General Properties

Tcl Console Messages Log Reports Design Runs

[Tue Feb 28 11:12:53 2023] Launched Lab00_sch_xup_and2_0_0_synth_1...

Run output will be captured here: C:/Xilinx/MyProject2023/Lab00_sch/Lab00_sch.runs/Lab00_sch_xup_and2_0_0_synth_1/runme.log

[Tue Feb 28 11:12:53 2023] Launched synth_1...

Run output will be captured here: C:/Xilinx/MyProject2023/Lab00_sch/Lab00_sch.runs/synth_1/runme.log

launch_runs impl_1 -jobs 8

[Tue Feb 28 11:14:18 2023] Launched impl_1...

Run output will be captured here: C:/Xilinx/MyProject2023/Lab00_sch/Lab00_sch.runs/impl_1/runme.log

Type a Tcl command here

Implementation Completed

Implementation successfully completed.

Next

Open implemented Design

Generate Bitstream

View Reports

Don't show this dialog again

OK Cancel

Implementation

Status: Complete

Messages: 6 warnings

Active run: impl_1

Part: xc7a35tsg324-1

Strategy: Vivado Implementation Defaults

Summary | Route Status

生成用于下载的bit文件

99% 11:16 2023/2/28

实验0.0：基本门电路的硬件实现



Lab0_sch - [C:/Xilinx/MyProject/2024/Lab0_sch/Lab0_sch.xpr] - Vivado 2019.1

File Edit Flow Tools Reports Window Layout View Help Quick Access

SYNTHESIZED DESIGN - synth_1 | xc7a35tcsg324-1

SYNTHESIZED DESIGN is out-of-date. Newer Synthesis results are available. Reload Close Design

Flow Navigator Run Simulation

RTL ANALYSIS Open Elaborated Design

SYNTHESIS Run Synthesis

Open Synthesized Design Constraints Wizard

Edit Timing Constraints

Set Up Debug

Report Timing Summary

Report Clock Networks

Report Clock Interaction

Report Methodology

Report DRC

Report Noise

Report Utilization

Report Power

Schematic

IMPLEMENTATION Run Implementation

Open Implemented Design

PROGRAM AND DEBUG Generate Bitstream

Device Constraints Internal VREF 0.6V

I/O Port Properties inv_S0

Name: inv_S0 Direction: IN

General Properties Configure Power

Tcl Console Messages Log Reports

Launch Runs

Launch the selected synthesis or implementation runs.

Launch directory: <Default Launch Directory>

Options

Launch runs on local host: Number of jobs: 6

Generate scripts only

Don't show this dialog again

OK Cancel

Implementation Status: Complete

Summary Route Status

Implementation Summary

All ports (5)

Scalar ports (5)

OUT	K3	34	LVC MOS33*	-	3.300	12	SLOW	NONE	FP_VTT_50
IN	T5	34	LVC MOS33*	-	3.300			NONE	NONE
IN	T3	34	LVC MOS33*	-	3.300			NONE	NONE
OUT	K2	35	LVC MOS33*	-	3.300	12	SLOW	NONE	FP_VTT_50
IN	R11	14	LVC MOS33*	-	3.300			NONE	NONE

实验0.0：基本门电路的硬件实现



Lab0_sch - [C:/Xilinx/MyProject/2024/Lab0_sch/Lab0_sch.xpr] - Vivado 2019.1

File Edit Flow Tools Reports Window Layout View Help Quick Access

SYNTHESIZED DESIGN - synth_1 | xc7a35tcsg324-1

Run Simulation

RTL ANALYSIS

Open Elaborated Design

SYNTHESIS

Run Synthesis

Open Synthesized Design

Constraints Wizard

Edit Timing Constraints

Set Up Debug

Report Timing Summary

Report Clock Networks

Report Clock Interaction

Report Methodology

Report DRC

Report Noise

Report Utilization

Report Power

Schematic

IMPLEMENTATION

Run Implementation

Open Implemented Design

PROGRAM AND DEBUG

Generate Bitstream

Device Constraints

Sources Netlist Device Constraints Project Summary

Internal VREF

0.6V

Drop I/O banks on voltages or the "NONE" folder to set/unset Internal VREF.

I/O Port Properties

Name: inv_S0

Direction: IN

General Properties Configure Power

Tcl Console Messages Log Reports Design

All ports (5)

Scalar ports (5)

OUT	K3	34	LVC MOS33*	-	3.300	12	SLOW	NONE	FP_VTT_50
IN	T5	34	LVC MOS33*	-	3.300			NONE	NONE
IN	T3	34	LVC MOS33*	-	3.300			NONE	NONE
OUT	K2	35	LVC MOS33*	-	3.300	12	SLOW	NONE	FP_VTT_50
IN	R11	14	LVC MOS33*	-	3.300			NONE	NONE

Bitstream Generation Completed

Bitsream Generation successfully completed.

OK Cancel

Implementation Summary Route Status

Complete

打开硬件设备管理器

?

Flow Navigator

Device Constraints

Project Summary

Overview Dashboard

Settings Edit

Project name: Lab0_sch

Project location: C:/Xilinx/MyProject/2024/Lab0_sch

Next

Open Implemented Design

View Reports

Open Hardware Manager

Generate Memory Configuration File

Don't show this dialog again

实验0.0：基本门电路的硬件实现



连接设备

The screenshot shows the Xilinx Vivado 2021.3 software interface. The left sidebar has sections for Flow Navigator, IMPLEMENTATION, PROGRAM AND DEBUG, and Open Hardware Manager. The main area is titled "HARDWARE MANAGER - unconnected" with a message "No hardware target is open. Open target". A red box highlights the "Auto Connect" button. Below it is a "Properties" panel. The bottom right shows the "Tcl Console" tab with the following text:

```
Read XDEF File: Time (s): cpu = 00:00:00 ; elapsed = 00:00:00.029 . Memory (MB): peak = 2201.383 ; gain = 0.000
Restored from archive | CPU: 0.000000 secs | Memory: 0.000000 MB |
Finished XDEF File Restore: Time (s): cpu = 00:00:00 ; elapsed = 00:00:00.029 . Memory (MB): peak = 2201.383 ; gain = 0.000
close_design
open_hw
```

Type a Tcl command here

Windows taskbar icons are visible at the bottom, along with system status and a date/time stamp: 21:06 2021/3/13.

实验0.0：基本门电路的硬件实现



國科大杭州高等研究院

Hangzhou Institute for Advanced Study, UCAS

A screenshot of the Xilinx Vivado 2021.3 software interface. The top menu bar includes File, Edit, Flow, Tools, Reports, Window, Layout, View, Help, and Quick Access. The left sidebar has sections for Flow Navigator, Report Clock Interaction, Report Methodology, Report DRC, Report Utilization, Report Power, Schematic, Implementation, Run Implementation, Open Implemented Design, Constraints Wizard, Edit Timing Constraints, Report Timing Summary, Report Clock Networks, Report Clock Interaction, Report Methodology, Report DRC, Report Utilization, Report Power, Schematic, and Program and Debug. Under Program and Debug, there are Generate Bitstream and Open Hardware Manager. A note at the bottom says '用指定的码流来编程序 程序烧录设备'. The main area shows the Hardware Manager window titled 'HARDWARE MANAGER - localhost/xilinx_tcf/Xilinx/1234-tulA'. It displays a tree view of hardware: 'localhost (1)' with 'xilinx_tcf/Xilinx/1234-tulA (Open)' expanded, showing 'xc7a35t_0 (1)'. A context menu is open over 'xc7a35t_0 (1)', with the 'Program Device...' option highlighted by a red box. Other menu items include 'Hardware Device Properties...', 'Verify Device...', 'Refresh Device', 'Show Bus Plot...', 'Add Configuration Memory Device...', 'Boot from Configuration Memory Device', 'Program BBR Key...', 'Clear BBR Key...', 'Program eFUSE Registers...', and 'Export to Spreadsheet...'. The bottom of the window shows a Tcl Console with some log messages and a text input field 'Type a Tcl command here'.

实验0.0：基本门电路的硬件实现



File Edit Flow Tools Reports Window Layout View Help Quick Access

Hardware Manager - localhost/xilinx_tcf/Xilinx/1234-tulA

There are no debug cores. Program device Refresh device

Hardware

Name Status

localhost (1) Connected

xilinx_tcf/Xilinx/1234-tulA (Open)

xc7a35t_0 (1) Programmed

XADC (System Monitor)

Program Device

Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file: _sch/Lab0_sch.runs/impl_1/Lab0_sch_wrapper.bit

Debug probes file:

Enable end of startup check

编程完成后，请检查启动结束（EOS）中的配置状态标志。

Program Cancel

启动编程

Tcl Console Messages Serial I/O Links Serial I/O Scans

Type a Tcl command here

INFO: [Labtoolstcl 44-406] Opening hw_target localhost:3121/xilinx_tcf/Xilinx/1234-tulA
: set_property PROGRAM.FILE [C:/Xilinx/MyProject/Lab0_sch/Lab0_sch.runs/impl_1/Lab0_sch_wrapper.bit] [get_hw_devices xc7a35t_0]
: current_hw_device [get_hw_devices xc7a35t_0]
: refresh_hw_device -update_hw_probes false [lindex [get_hw_devices xc7a35t_0] 0]
INFO: [Labtools 27-1434] Device xc7a35t (JTAG device index = 0) is programmed with a design that has no supported debug core(s) in it.

21:08 2021/3/13

实验0：基本门电路的硬件实现



■ 实验效果

- 按键S0按下，LD2_0熄灭，按键S0松开，LD2_0点亮
- DIP8：
 - SW0和SW1均为ON，LD1_0点亮
 - SW2或SW3一个为ON，LD1_2点亮
 - SW4和SW5不同，LD1_4点亮
 - SW6和SW7相同，LD1_6点亮



实验0：基本门电路的硬件实现



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

二选一复用器

$$Z = \sim S \& X + S \& Y$$



THANKS



國科大杭州高等研究院
Hangzhou Institute for Advanced Study, UCAS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第二讲 (2.1 FPGA概述)

2025年3月11日



1. FPGA的发展历史

2. FPGA与其他可编程器件的区别与联系

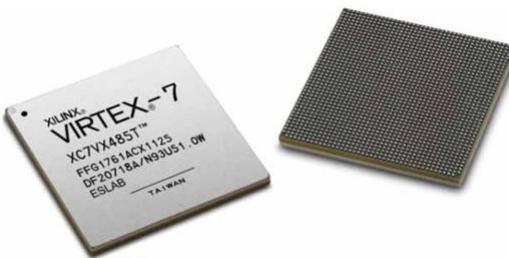
3. FPGA的应用概述

4. FPGA的硬件架构



什么是FPGA?

- FPGA英文Field Programmable Gate Array，即现场可编程门阵列
- FPGA本质是一个可定制的逻辑电路
- 作为专用集成电路ASIC领域中的一种半定制电路
- 解决了定制电路的不足
- 克服了原有可编程器件门电路数有限的缺点





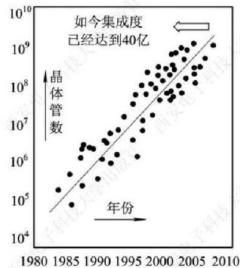
什么是FPGA?

- Synopsys公司高级营销经理Joe Mallett说：“如果每家汽车公司都从FPGA入手，那么一开始就要计算在什么时候将其转换为 ASIC 才有意义”
- 专用集成电路（ASIC）的开发成本很高，而且需要很长时间才能推向市场
- 通常情况下，FPGA 的销售预期是先在市场上站稳脚跟，然后再降低成本，但很多时候，这些量并没有完全实现，FPGA 仍然留在原地

FPGA概述-FPGA发展史



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

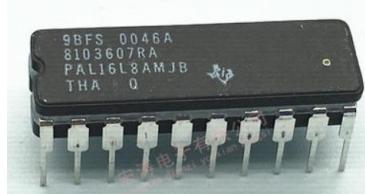


- 集成电路源于上世纪60年代（小规模集成电路SSI,10门以下），集成电路的发展先后经历了：
- 中规模（10-100门，70年代，微处理器、通用逻辑电路以及存储器）
 - 大规模（100-1万门，80、90年代，MCU,CPU）
 - 超大规模（1万-10万门，GPU等）
 - 极大规模（10万门以上）



NVIDIA单芯显卡GeForce GTX Titan
晶体管数量达到80亿

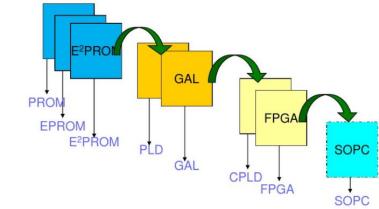
- 1970年以PROM的形式进入人们的视野第一款可编程逻辑器件（PLD）；
PROM结构接口少
- 发展出可编程逻辑阵列PAL和通用阵列逻辑GAL、反熔丝技术、EPROM和EEPROM技术，但**结构功能仍然相对简单**
- Altera公司于1984年发明了基于CMOS和EPROM技术相结合的CPLD：可**胜任复杂性较高、速度也较快的逻辑功能**
- Xilinx创始人之一Ross Freeman于1985年发明了FPGA，开启了可编程逻辑的“高速”发展时代



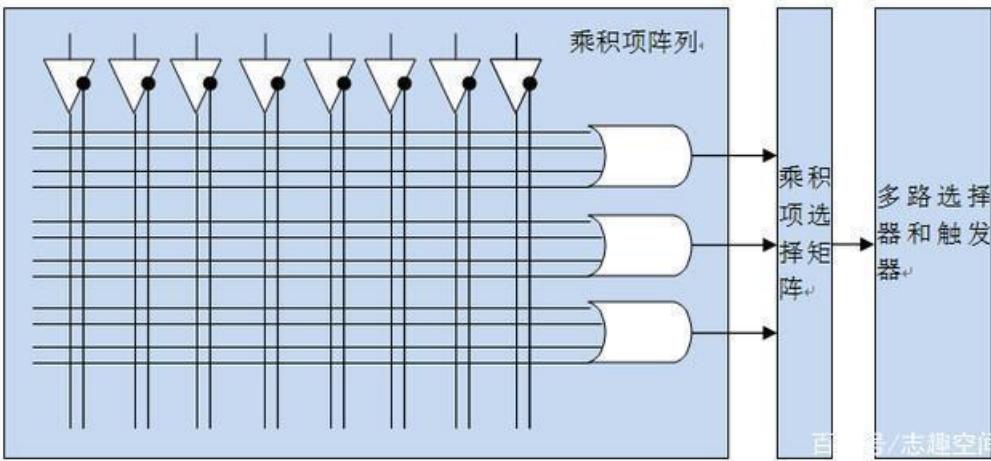
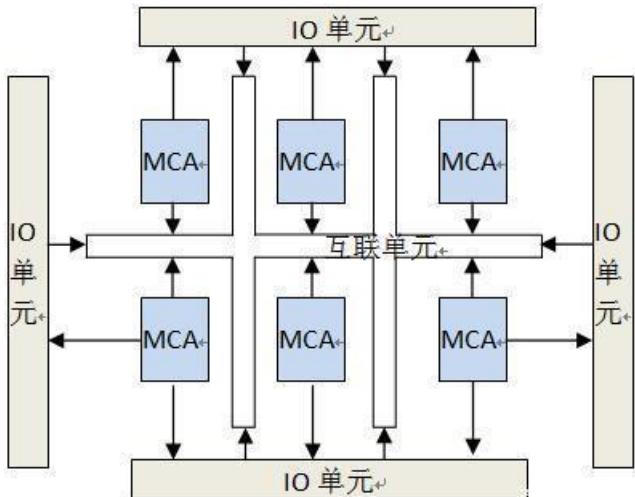
已经绝迹的PAL/GAL



曾经火过一时的CPLD

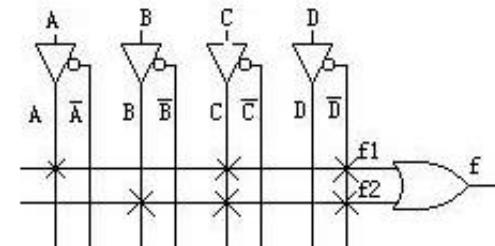


FPGA概述-FPGA发展史



CPLD

- CPLD由I/O单元, MCA(可编程逻辑宏单元阵列), 互联单元等部分组成
- 任何组合逻辑电路均可以化简为与或式
- 乘积项阵列中每个交叉点代表与逻辑, 乘积项选择代表或逻辑
- CPLD有较多的组合逻辑资源, CPLD不需要配置芯片



$$f = f_1 \mid f_2 = A \& C \& !D \mid B \& C \& !D$$

FPGA概述-FPGA发展史



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

FPGA大放异彩

- Ross Freeman (1948-1989) 是Xilinx公司的共同创始人之一，荣登2009年美国发明家名人堂 (2009年时一共405人入选)
- Freeman先生发明的FPGA是一块全部由“**开放式门**”组成的计算机芯片，Freeman按照摩尔定律（晶体管数量每两年翻一番）准确推测，晶体管成本将随时间推移稳步下降，**低成本、高度灵活的 FPGA 将成为各种应用中定制芯片的替代品**
- “当他描述完FPGA的概念之后，我的第一个念头就是，疯了！这是史上对晶体管最不靠谱的浪费了！”—— Bill Carter。在晶体管**非常宝贵的年代**，大量使用晶体管的想法有些**匪夷所思**
- 他的远见卓识和创造热情催生了可编程芯片，这项技术不仅影响着之后电子产业的发展，还推动赛灵思的客户不断设计出创新型终端产品，从而使人们的生活质量不断得到提高。
- 近年来，可编程器件的龙头老大Xilinx和Altera更是相继推出了硬核**CPU+FPGA**的产品，此举大有单芯片横扫千军的架势



FPGA发明人 Ross Freeman



野心十足的ARM+FPGA架构



● FPGA与ASIC

- ASIC，即专用集成电路（Application Specific Integrated Circuit）的简称
- ASIC的功能和性能都相对固定，它是为了专一功能或专一应用领域而生，通常无法进行任何的功能和性能的修改
- FPGA是“**可反复编程的逻辑器件**”，与ASIC最大的区别就是FPGA具有数字电路的可重复编程特性，而ASIC不具有

FPGA概述



● FPGA、ARM和DSP

- **ARM:** 侧重于控制和传输，通常也包含丰富的人机交互功能的处理器
- **DSP:** 为高速数据的实时采集和传输、复杂的运算处理应用而定制化的处理器
- **FPGA:** 覆盖各种ARM和DSP无法满足的应用，更趋向于小批量、定制化、
实时性要求有很高的应用，理论上讲FPGA可以实现ARM和DSP的所有功能
- ARM、DSP均是基于“指令”对硬件的控制，工作方式是“指令的执行”，
FPGA本质上是纯硬件（变成CPU电路之后的二次编程除外）
- 三者某种意义上是互补的，甚至一些芯片上存在二者共生的现象，如Xilinx
的ZYNQ中有ARM也有FPGA



ARM处理器



DSP处理器



FPGA器件



● FPGA与GPU

● 峰值性能，GPU远高于FPGA

能够集成上千个内核的GPU，高度架构优化，关键路径可手动调整，绝对性能可接近工艺极限

FPGA设计资源受限，时钟频率受制于目标设计的布局布线，绝对性能有一定瓶颈，不及GPU

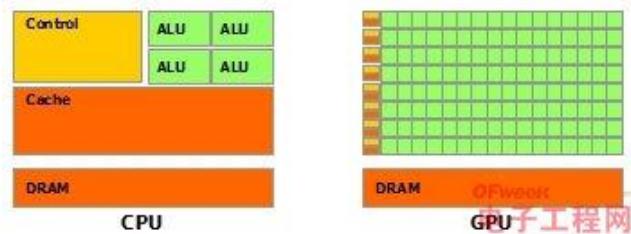
● 灵活性，FPGA远好于GPU

GPU产品化之后，硬件架构固定，无法满足一些小众算法或新生算法对硬件架构的最优化“调整”

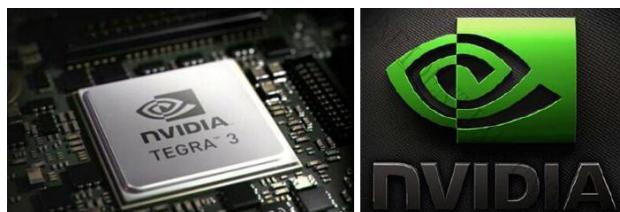
FPGA的可编程特性，各种不同的硬件架构最优化需求都能得到满足，一定程度上弥补绝对性能的不足

● 功耗，GPU的绝对功耗远大于FPGA

在同等效率情况下，经过架构优化的FPGA功耗也远低于GPU



绿色的是计算单元，橙红色的是存储单元，橙黄色的是控制单元



FPGA技术所能够带来的潜在优势



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

● 灵活性

可重编程，可定制

易于维护，方便移植、升级或扩展

降低成本，加速产品上市时间

支持丰富的外设接口，可根据需求配置

● 并行性、低延迟

更快的速度、更高的带宽

低延迟特性，满足实时处理的需求

● 集成性

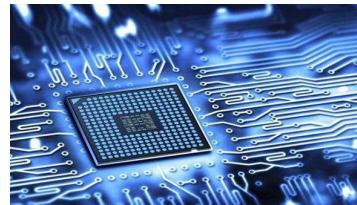
更多的接口和协议支持

可将各种端接匹配元件整合到器件内部

有效降低BOM成本

单片解决方案，可以替代很多数字芯片

减少板级走线，有效降低布局布线难度



FPGA技术的局限性



- **绝对性能受限：**在某些性能上，FPGA可能比不上专用芯片，或者至少在稳定性方面，
FPGA要逊色一些
- **功耗相对较高：**相比特定功能、应用集中的ASIC，使用FPGA实现相同的功能可能产
生更高的功耗（相比GPU，则单位性能功耗要低很多）
- **成本较高：**尤其在消费级领域，大批量产品中ASIC芯片具有成本优势
- **设计复杂性高：**FPGA设计复杂性和难度可能会给产品的开发带来一场噩梦





◆ 是否使用FPGA技术来实现目标产品？

- 产品性能
- 开发周期
- 可升级性
- 实现成本
- 技术可行性
- 其他限制因素



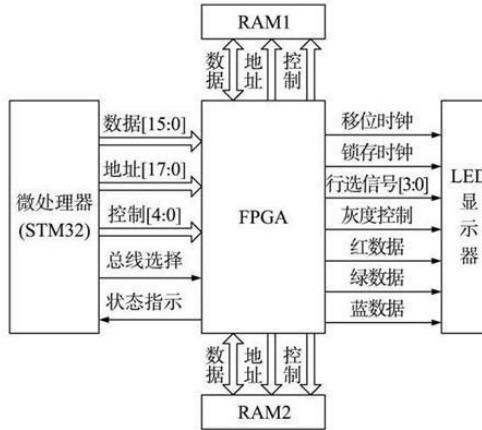
FPGA概述—应用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

● 逻辑粘合

- 如一些嵌入式处理常常需要**地址或外设扩展**, CPLD器件尤其适合
- 目前已经少有项目会选择一个FPGA器件专门用于逻辑粘合的应用,
但是在已经使用FPGA器件中顺便做些逻辑粘合的工作是非常普遍的

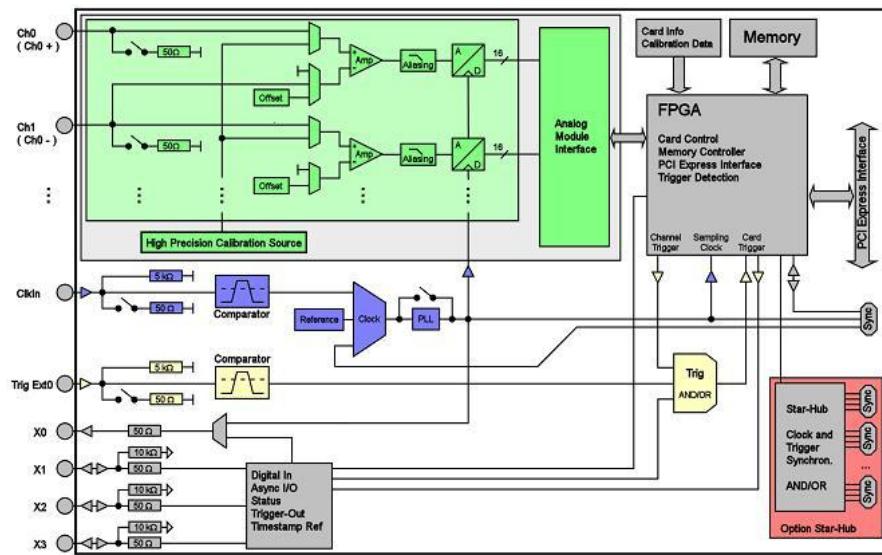
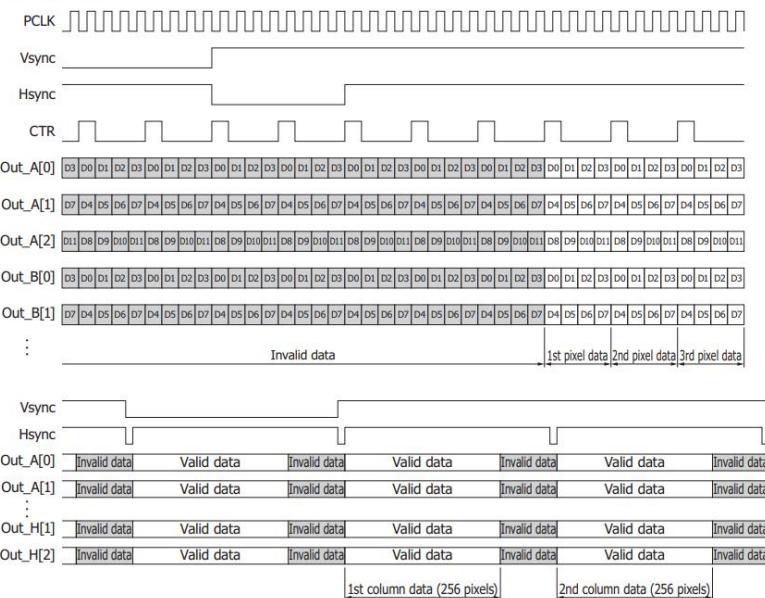


FPGA概述—应用



- 高速信号采集和处理
- 如高速A/D前端或图像前端的采集和预处理
- 持续升温的机器视觉应用几乎是无一例外的使用了FPGA器件

Timing chart



FPGA概述—应用



國科大杭州高等研究院

Hangzhou Institute for Advanced Study, UCAS

● 协议实现

- 如更新较快的各种有线和无线通信标准、广播视频及其编解码算法、各种加密算法等
 - 诸如此类小批量、定制化、更新换代频繁的应用使用FPGA比ASIC更具有竞争力



FPGA概述—应用

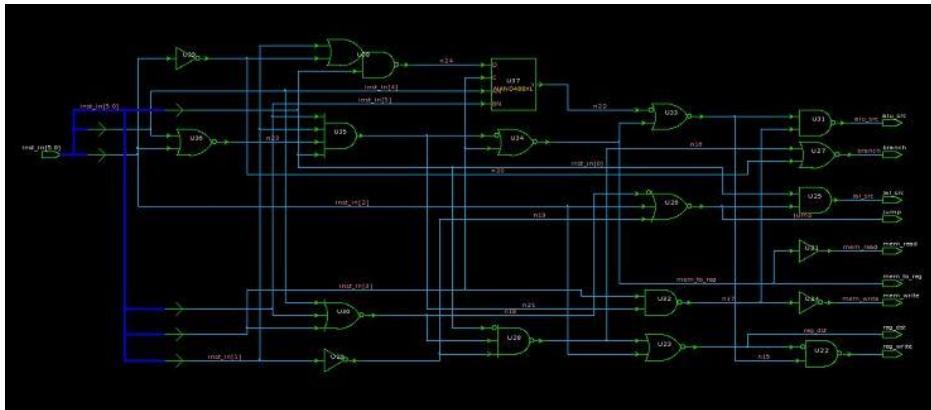


國科大杭州高等研究院

Hangzhou Institute for Advanced Study, UCAS

● 各种原型系统验证

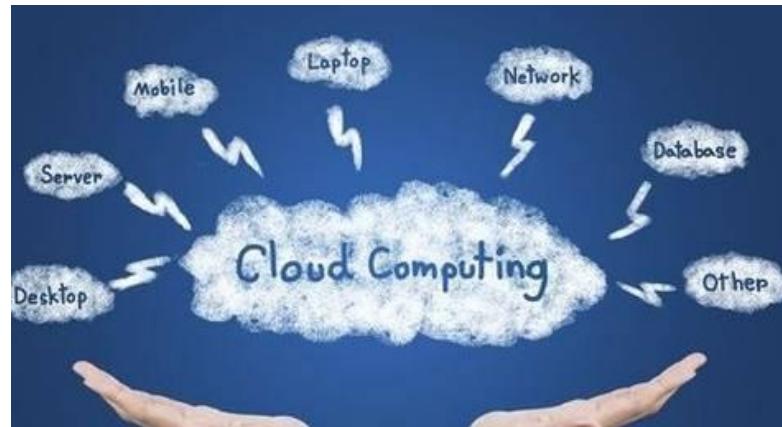
- **FPGA支持丰富的接口协议标准，可定制性强**
 - **芯片原厂使用FPGA搭建芯片的验证**





● 并行计算（算法实现）

- 传统的CPU计算受限于其串行顺序处理的架构，已经很难适应今天的云计算和数据中心对大数据运算的需求；而FPGA与生俱来的**并行性与灵活可编程特性**是其进入高速运算领域的一个优势
- GPU虽然一直是并行处理的主流方案，但也受限于极高的成本和功耗代价；相比之下，单位功耗性能是GPU3~4倍的FPGA则大有**分庭抗礼**之势



FPGA概述—应用



● FPGA与大数据、AI

- 通用处理器（CPU）的摩尔定律已进入暮年，而机器学习和Web服务的规模却在指数级增长
- CPU、GPU都属于冯·诺依曼结构，指令译码执行、共享内存，FPGA无指令、无需共享内存
- 矩阵运算、图像处理、机器学习、压缩、搜索排序等计算密集型任务由CPU offload给FPGA执行
- 可被重新编程来执行新类型的计算任务
- FPGA的流水线并行和数据并行，高吞吐量和低延时
- 2014年微软在顶会ISCA上发表“*A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services*”论文，标志着FPGA第一次在互联网/软件公司的大型数据中心得到实质性应用
- 亚马逊AWS、阿里云、腾讯云、百度智能云数据中心都部署了FPGA硬件加速



微软Catapult团队



微软第一代FPGA加速板卡



阿里FPGA云服务器



百度FPGA云服务器

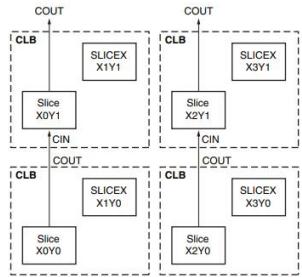
FPGA概述-架构



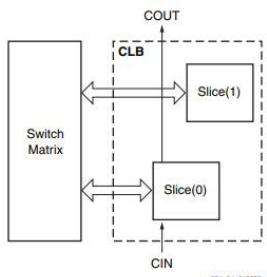
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



最大的不同?

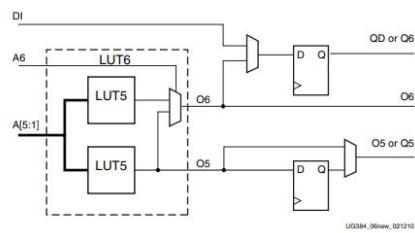


Row and Column Relationship
between CLBs and Slices



Arrangement of Slices within the CLB

- 可配置逻辑块CLB/Slice
- 时钟管理模块CMT
- 存储器RAM
- 数字信号处理模块DSP
- 专用模块



LUT6

https://china.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf
<https://www.intel.com/content/www/us/en/products/programmable/fpga/cyclone-v/features.html>

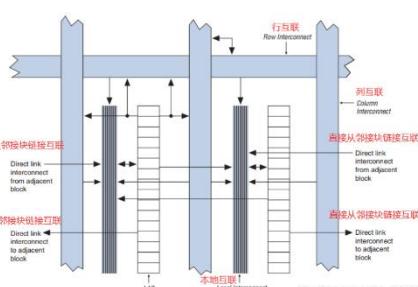
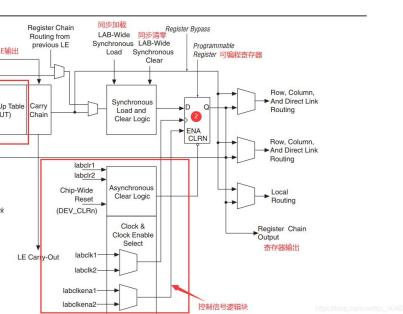


Figure 2-1. Cyclone IV Device LEs



- 逻辑阵列模块LAB/LEs
- 锁相环PLL
- 存储器RAM
- 数字信号处理模块DSP



◆ 数字电路中的0和1

◆ 逻辑0和1

- ◆ 我们的工作和生活几乎已经被各种无孔不入的“数字化”设备所充斥着
- ◆ 电脑、手机、各种娱乐设备所面对的图像、影音、文字资料，皆是以0和1的符号来存储、传输和处理的.....

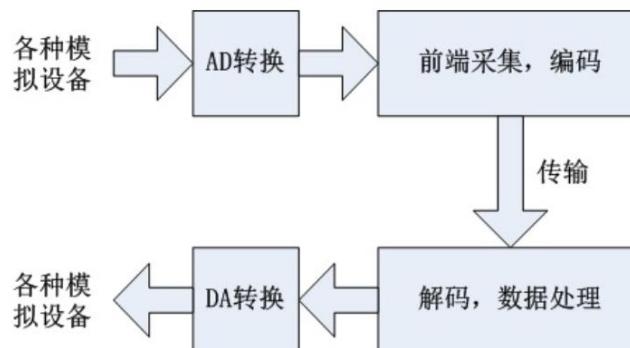
◆ 模拟和数字转换

- ◆ 数字电路在现实世界中还是要依靠模拟作最终的载体
- ◆ 设备大都需要经过模拟、数字再到模拟的转换过程

◆ 数字逻辑的关系和计算

- ◆ 非门、与门、或门、与非门、或非门、异或门、同或门

在上一节课中已部分实现





◆ 逻辑门电路的使用

◆ 逻辑门电路与二进制运算—1位加法器

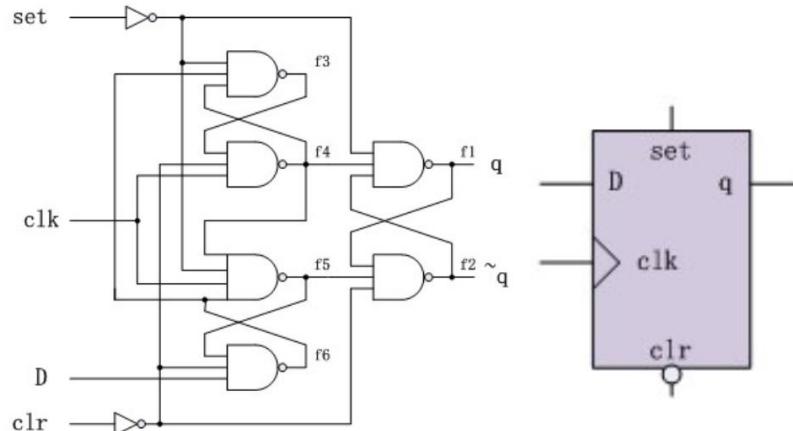
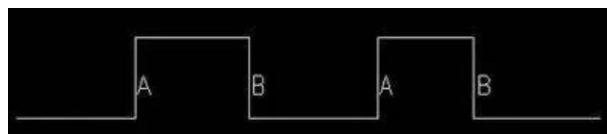
- ◆ x 与 y 相加，结果为 z ，进位为 c
- ◆ $z = x \wedge y, c = x \& y$

x	0	0	1	1
y	0	1	0	1
z	0	1	1	0
c	0	0	0	1



◆ 逻辑门电路与触发器-D触发器

- ◆ 时钟信号为方波信号，下图A为上升沿，B为下降沿
- ◆ 每当一个上升沿（从0到1）来临，输入信号D输出到q
- ◆ set（置位）当它为1时，输出q一定为1，无论输入什么
- ◆ clr（复位）当它为1时，输出q一定为0，无论输入什么
- ◆ D触发器可以理解为存储器件





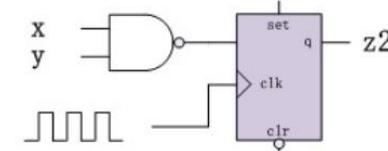
◆ 时序逻辑与组合逻辑

◆ 组合逻辑

- ◆ 不含有任何存储比特信号的电路
- ◆ 输出只与当前电路的输入有关
- ◆ 立即反应当前输入状态
- ◆ 时序比较难保证
- ◆ 适合简单的电路



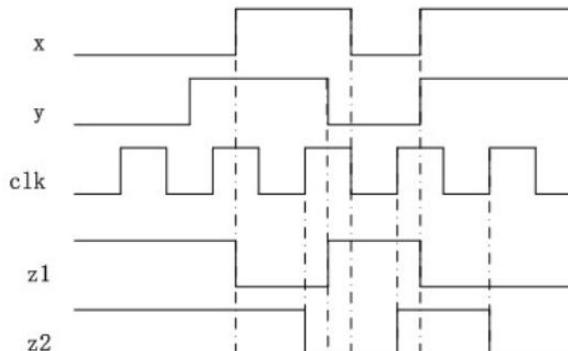
组合逻辑



时序逻辑

◆ 时序逻辑

- ◆ 一定有用于存储比特信号的电路
- ◆ 输出不仅与当前输入值有关，和电路的原有状态相关
- ◆ 必须在时钟上升（下降）触发后输出新值
- ◆ 容易达到时序收敛，时序逻辑更可控
- ◆ 能够胜任大规模的逻辑电路



FPGA概述-架构



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

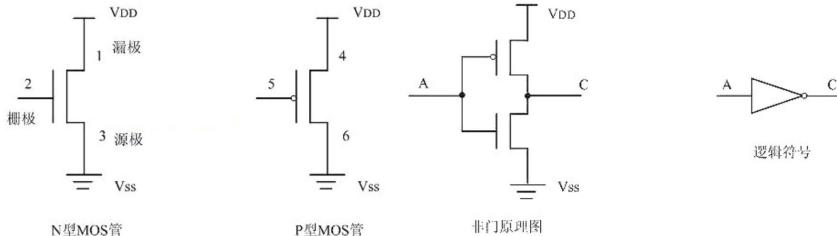
◆ FPGA中逻辑门的硬件实现

◆ 逻辑门电路与晶体管

- ◆ 所有逻辑门电路都可以由PMOS和NMOS实现
- ◆ 非门可以由一个PMOS和一个NMOS实现
- ◆ 其他逻辑门均可由晶体管实现
- ◆ 但是在FPGA中不是使用晶体管实现的

◆ FPGA中的查找表（LUT）

- ◆ 输入作为地址，输出结果为对应地址的内容
- ◆ 右图为与门的LUT的实现
- ◆ FPGA中可使用LUT实现各种逻辑门



输入x	0	1	0	1
输入y	0	0	1	1
地址	0	1	2	3
输出z	0	0	0	1

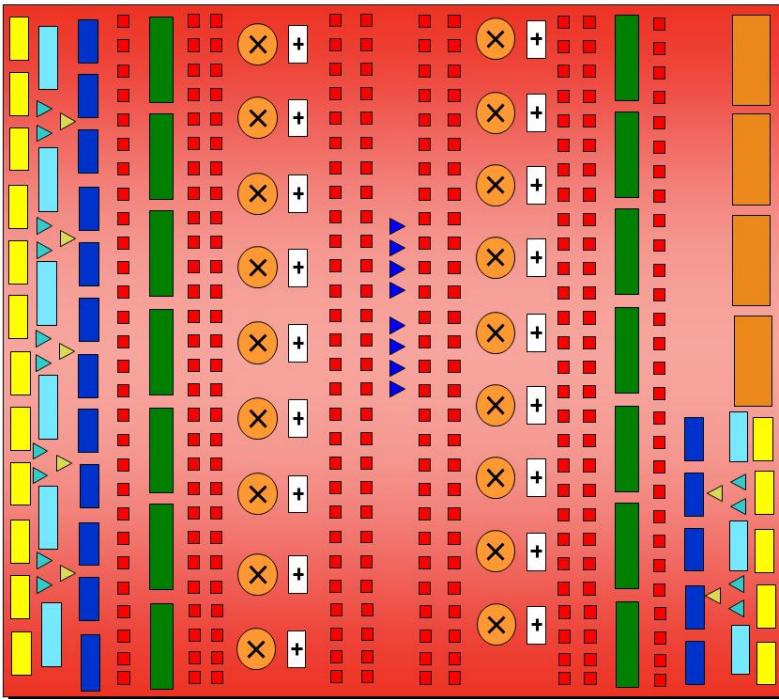
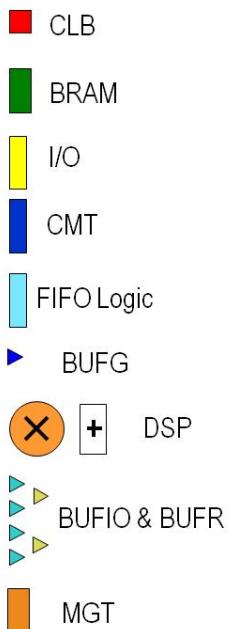
地址0	0
地址1	0
地址2	0
地址3	1

FPGA概述-架构



Xilinx 7系列FPGA架构

- CLB (configurable logic blocks)
- BRAM (Block RAM)
- IO (Input & Output)
- CMT (clock management tile)
- FIFO (First In First Out)
- BUFG (全局时钟缓冲器)
- DSP (数字信号处理器)
- BUFIO (IO时钟网络, 仅用于IO Block)
- BUFR (区域时钟网络)
- MGT (Multi-gigabittransceiver)





Xilinx 7系列FPGA架构

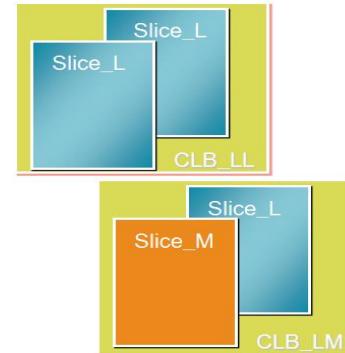
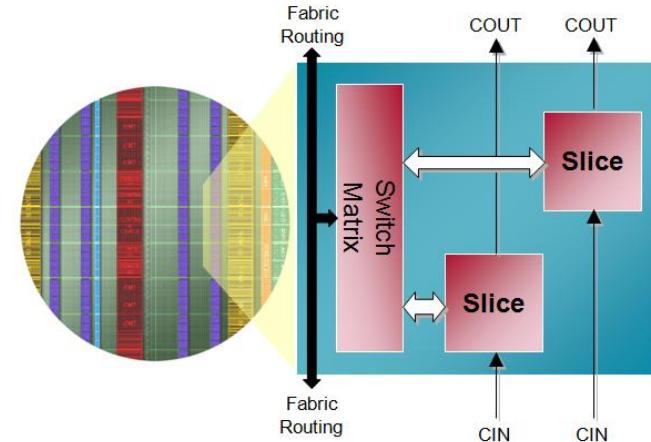
7 Series : User Guides	
	7 Series FPGAs Memory Resources User Guide
	7 Series FPGAs Configurable Logic Block User Guide
	7 Series FPGAs SelectIO Resources User Guide
	7 Series FPGAs Clocking Resources User Guide
	7 Series FPGAs Configuration User Guide
	7 Series FPGAs DSP48E1 Slice User Guide
	7 Series FPGAs GTX/GTH Transceivers User Guide
	7 Series FPGAs GTP Transceivers User Guide
	7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide

FPGA概述-架构



CLB

- FPGA中**最重要的设计资源**, 实现**组合逻辑, 触发器**
- 每个CLB包含**两个Slices**
- Slice_L: 只能用于**逻辑**
- Slice_M: 除逻辑功能外, 还可用于**存储及移位寄存器**
- CLB_LL: 包含**两个Slice_L**
- CLB_LM: 包含一个Slice_L和一个Slice_M



FPGA概述-架构



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Slice_M
Slice_L

- 4个六输入LUT
- 8个触发器
- 多路复用器
- 进位逻辑
- Slice_M可配置成分布式RAM

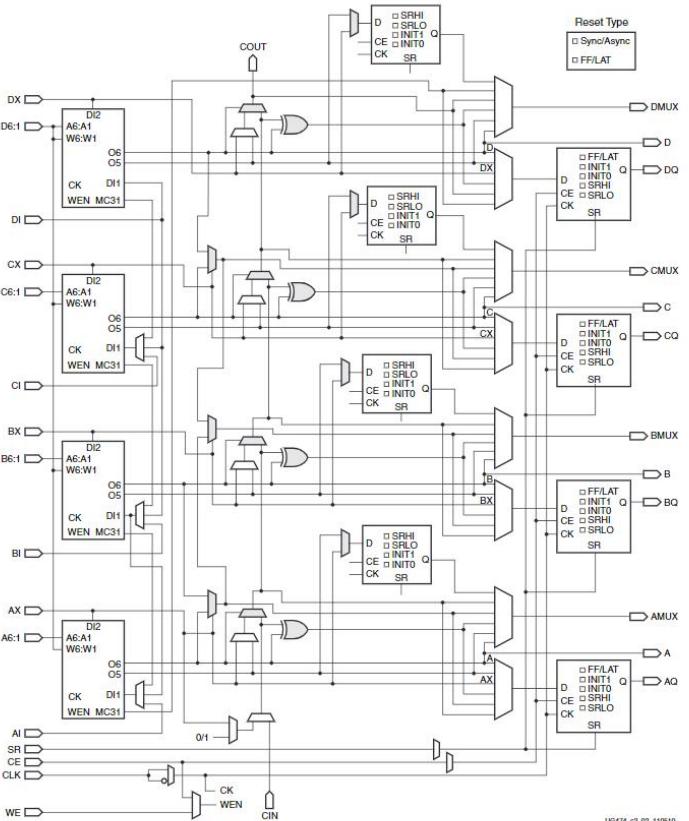


Figure 2-3: Diagram of SLICEM

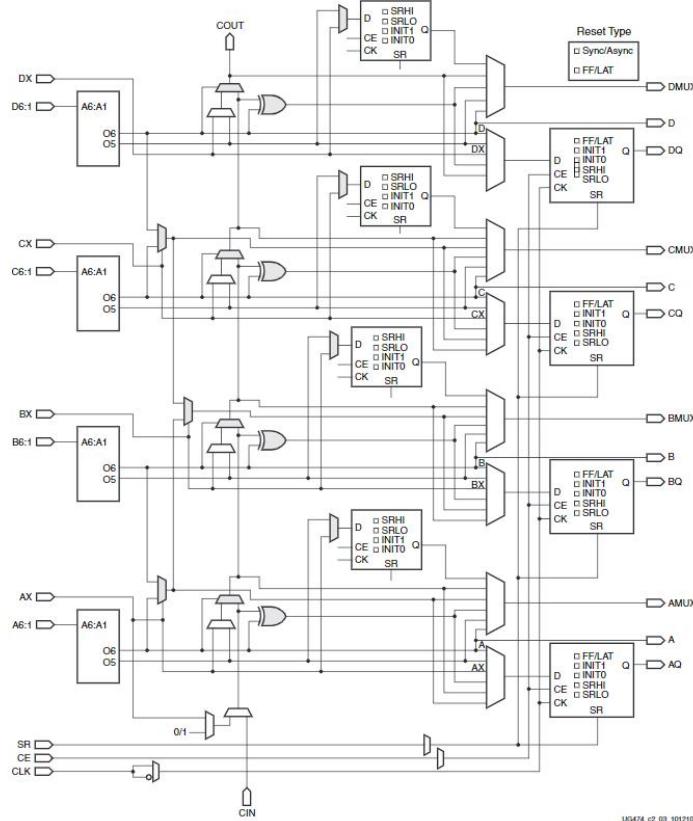


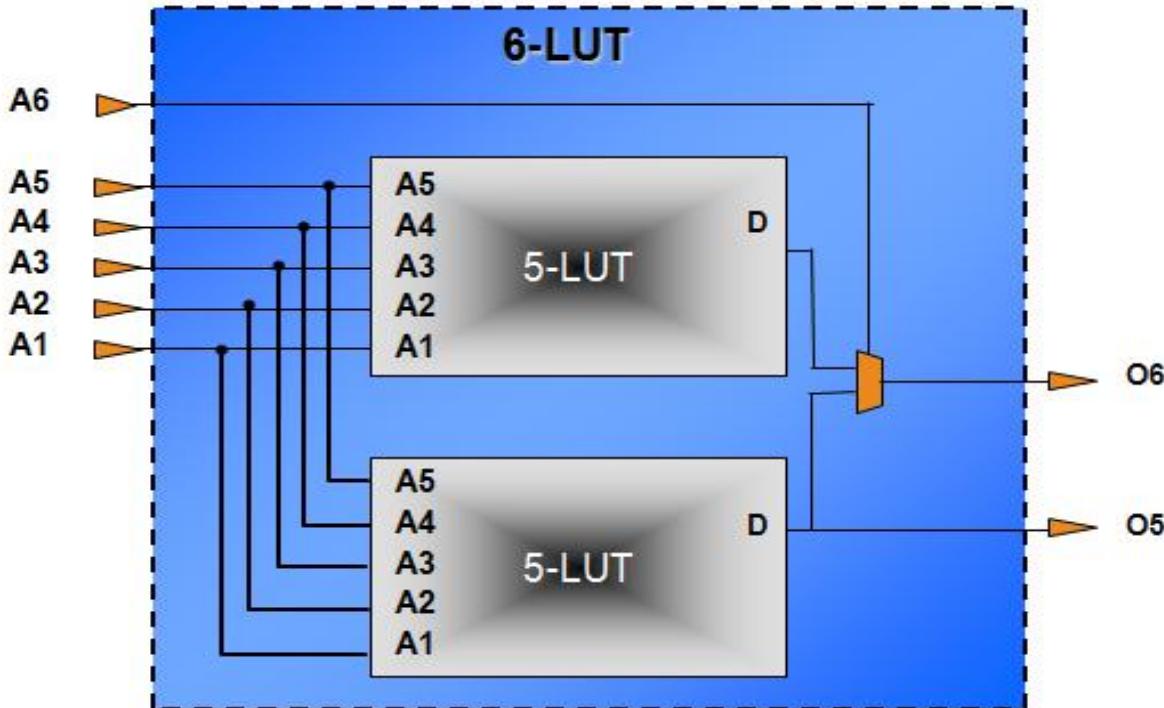
Figure 2-4: Diagram of SLICEL

FPGA概述-架构



LUT

- 实现任意6输入的组合逻辑
- 可配置成两个5输入LUT



FPGA概述-架构



I/O

- 可配置成单端或差分
- ILOGIC即输入信号处理逻辑，由数据选择器和IDDR触发器构成，IDDR触发器可以双沿捕获输入数据也可以拆分成普通单沿触发器
- IDELAY被称为信号延迟模块，它的作用就是把信号延迟一段时间
- ODELAY是用作输出信号的延迟

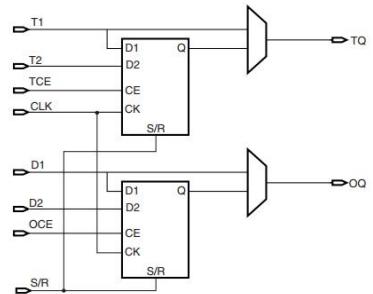
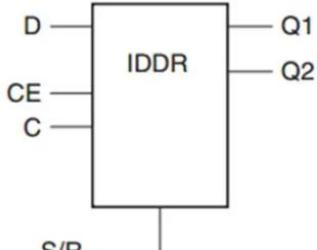
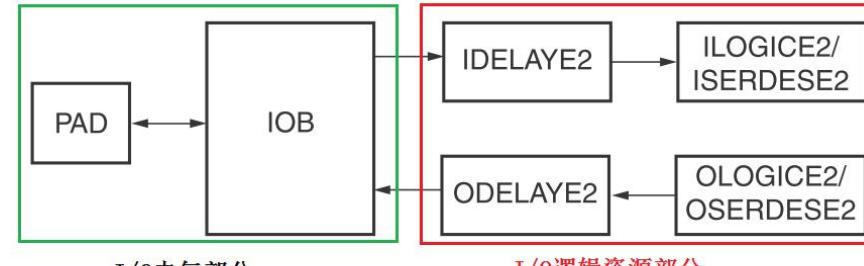
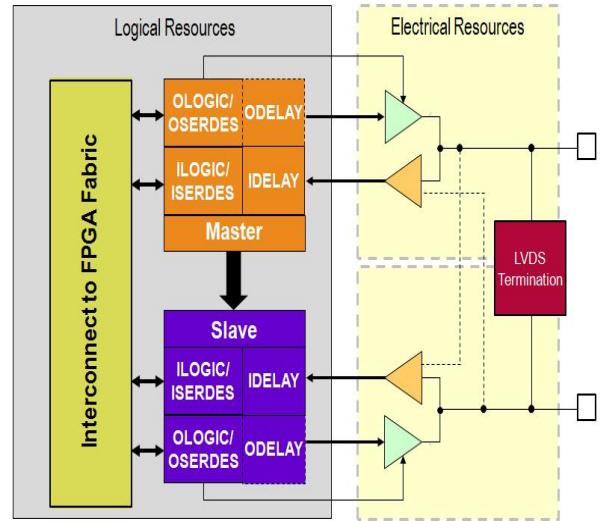


Figure 2-17: OLOGIC Block Diagram



I/O电气部分

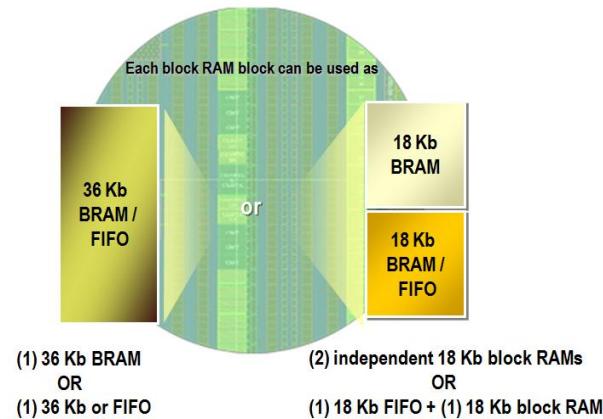
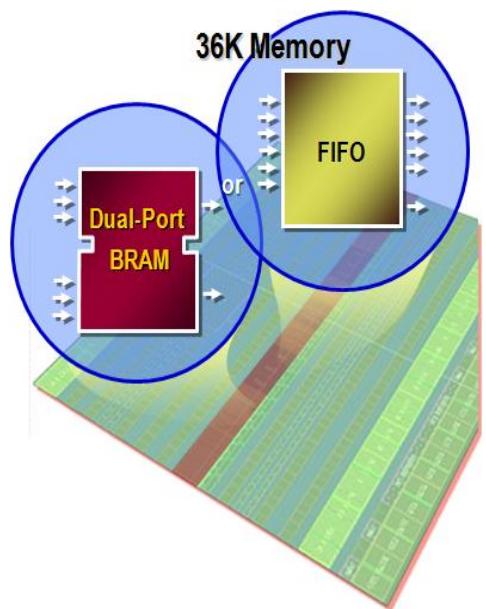
I/O逻辑资源部分

I/O触发器



Block RAM and FIFO

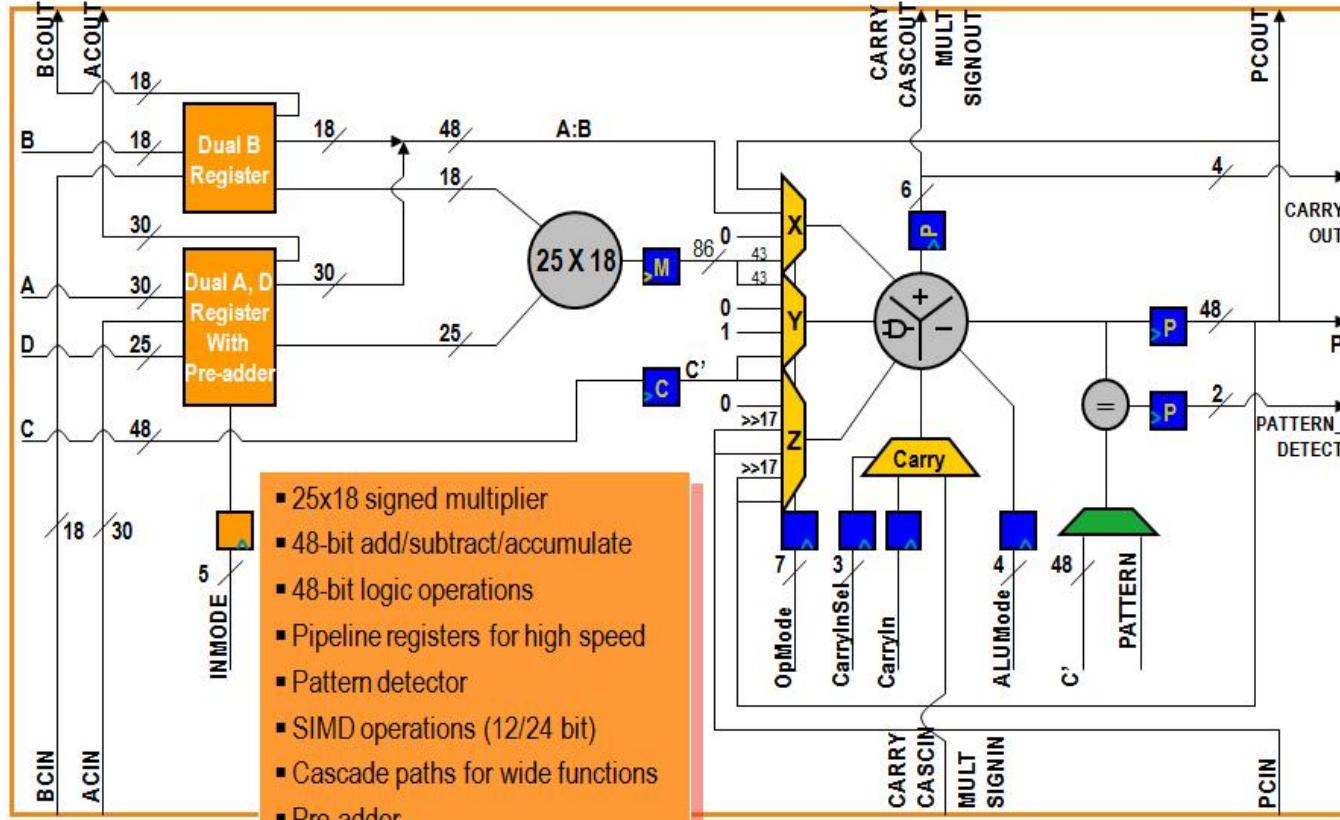
- 每一个单元均包括BRAM和FIFO
- 36Kb BRAM或36Kb FIFO
- 2 x 18Kb BRAM或 18Kb FIFO+18Kb BRAM



FPGA概述-架构



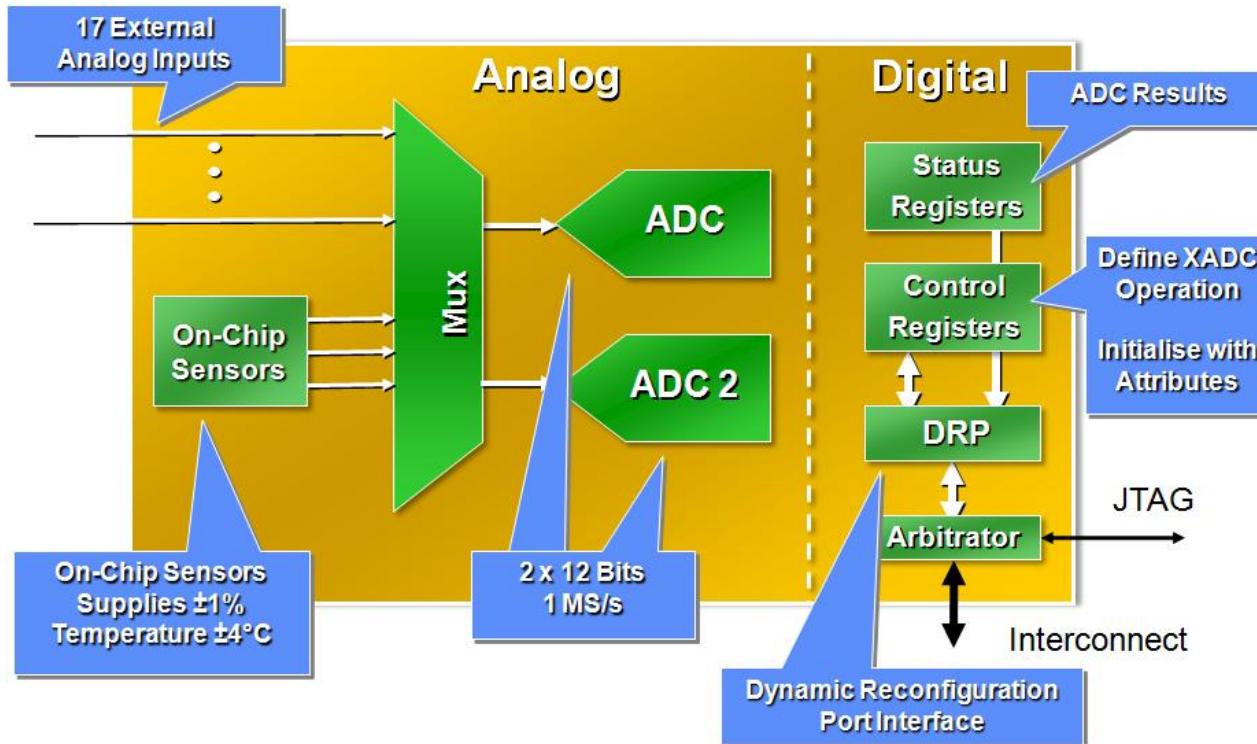
DSP



FPGA概述-架构



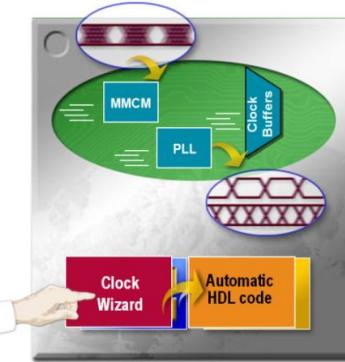
XADC





CMT

- 包括一个MMCM(Mixed-Mode Clock Managers)和一个PLL(Phase Locked Loop)
- 所有的时序电路都需要至少一个时钟信号
- 每一个I/O Bank都有专用时钟输入引脚，均可配置成单端或差分输入



7 Series FPGAs Clocking Resources

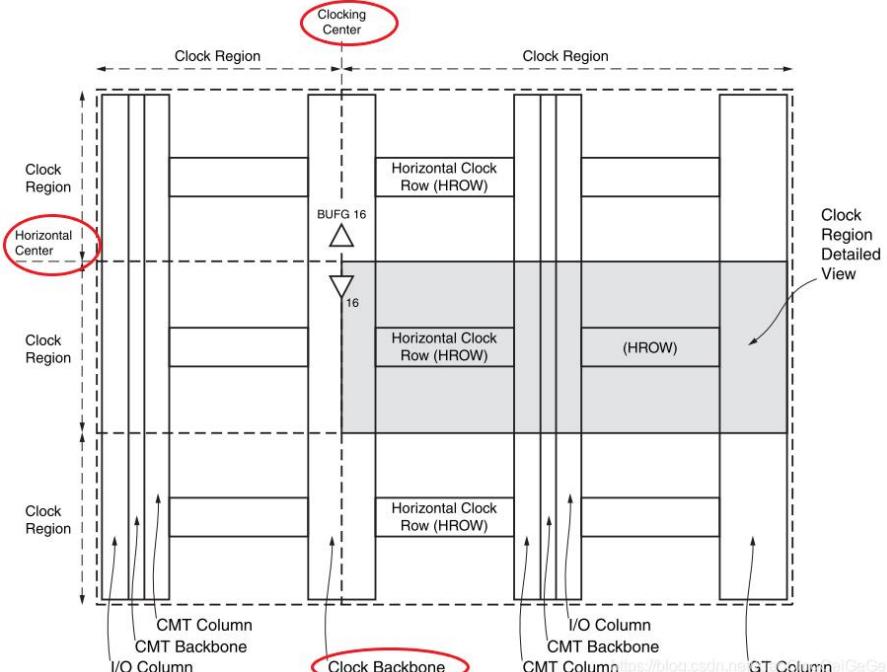
User Guide

FPGA概述-架构



CMT-时钟布线资源

- **Clock Region**: FPGA内部分成了很多个**时钟区域**, 每个时钟区域包含一个CMT
- **Horizontal Center**: FPGA被Horizontal Center分成上下两个部分, 每个部分包含16个BUFG
- **Clock Backbone**: 全局时钟线的主干道, 将FPGA分成了左右两部分, 所有的全局时钟布线均要从此经过。
- **HROW**: 水平时钟线, 从水平方向贯穿每个时钟区域的中心区域, 将时钟区域分成上下完全一致的两部分。全局时钟线进入每个时钟区域的逻辑资源时, 必须经过水平时钟线。
- **CMT Backbone**: 对于相邻时钟区域的时钟布线, 可以不使用珍贵的全局时钟网络, 而使用每个时钟区域都包含的CMT Backbone通道。
- **I/O Column**: 外部信号/时钟输入管脚。
- **CMT Column**: 每个时钟区域都包含一个CMT, 一个CMT由一个MMCM和一个PLL组成。
- **GT Column**: 内含高速串行收发器。



FPGA实际上就是被分成很多个大小一样时钟区域, 每个时钟区域既可单独工作又可通过全局时钟Clock Backbone统一工作, 同时水平相邻的时钟区域又可通过HROW来统一工作, 上下相邻的时钟区域又可通过CMT Backbone统一工作

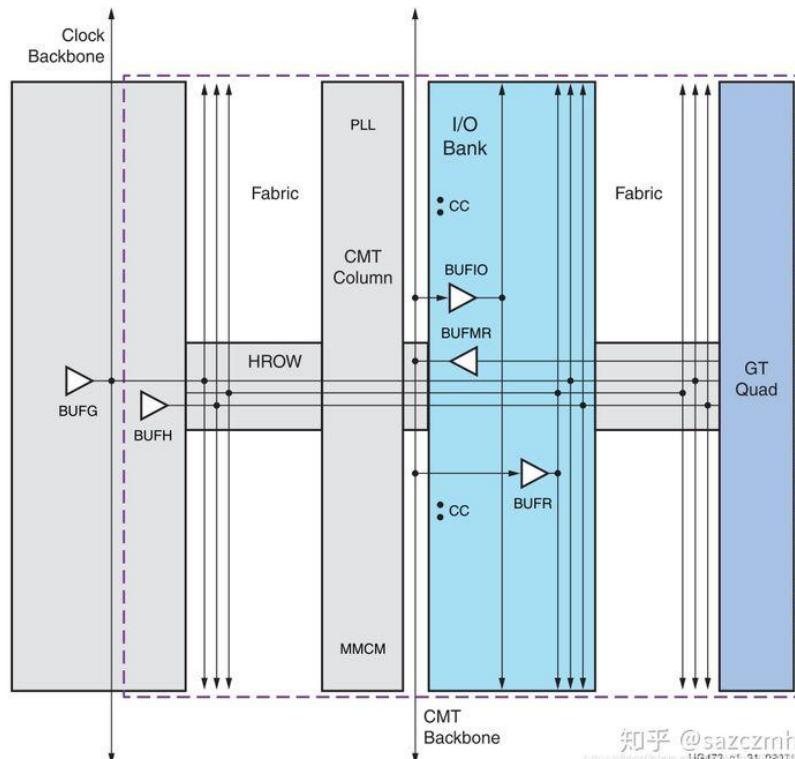
FPGA概述-架构



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

CMT-Clock Region

- **BUFG**即为全局时钟缓冲器
- **BUFH**即为水平时钟缓冲器，它相当于一个功能受限的BUFG，其输出时钟只能通过HROW在左右相邻的时钟区域内工作。
- **BUFIO**即为IO时钟缓冲器，其输出时钟只能作用在一个时钟区域的IO寄存器处，无法在FPGA内部逻辑使用。
- **BUFR**即为区域时钟缓冲器，其输出只能作用在一个时钟区域，相当于BUFH的阉割版。
- **BUFMR**即为多区域时钟缓冲器，其输出作用在本时钟区域，还可以通过CMT Backbone作用在上下相邻两个时钟区域。
- **MMCM、PLL**即时钟管理模块，用来消除时钟的延迟、抖动以及产生各种不同频率的时钟。
- **CC**即为外部时钟输入管脚，其管脚在内部可以连接到BUFG、BUFR、BUFIO、BUFH、MMCM、PLL等

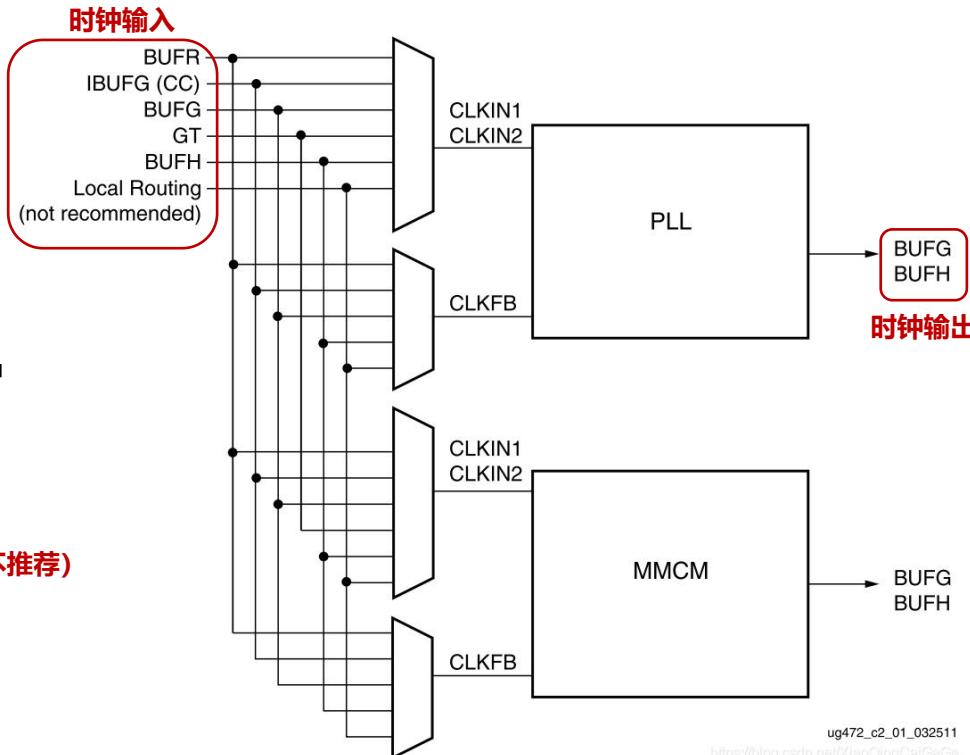


FPGA概述-架构



CMT

- 频率综合：将外部输入的固定频率时钟调理成多路可调节频率的时钟
- 去抖动（内部带有滤波电路）
- 去偏斜
- 时钟输入：BUFR、CC、BUFGR、GT、BUFH、内部走线（允许但不推荐）
- 时钟输出：BUFGR、BUFH



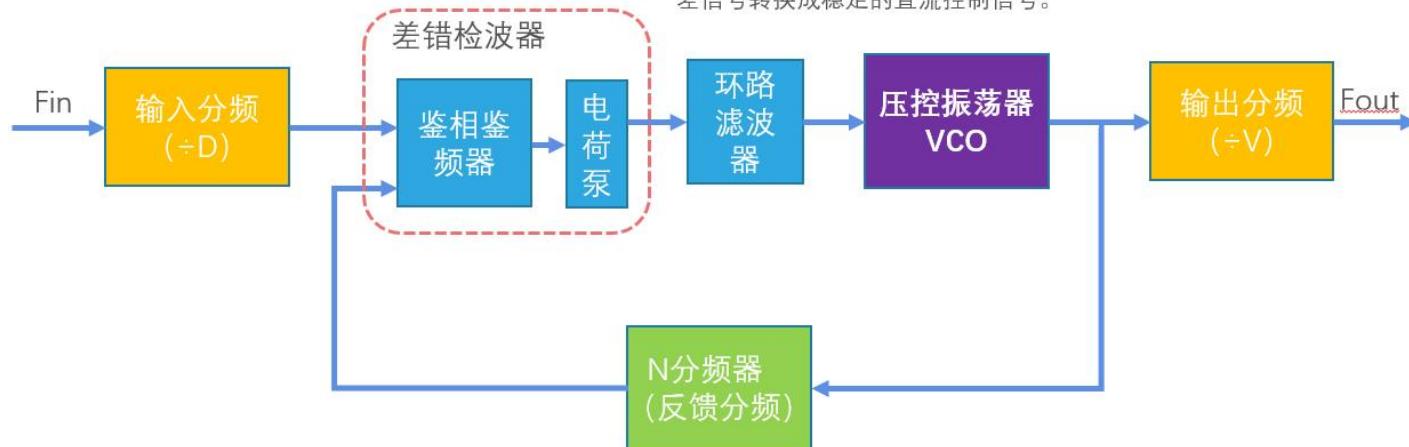


CMT

锁相环架构及工作过程

差错检波器：当两个输入信号的相位及频率相等时，检波器中的比较结果会处于稳定状态，此时误差为常数，环路处于“锁定”条件。

环路滤波器：具有低通滤波器特性，将差错检波器输出的误差信号进行滤波（滤除高频成分、噪声以及鉴相器中包含的交流分量），并将误差信号转换成稳定的直流控制信号。



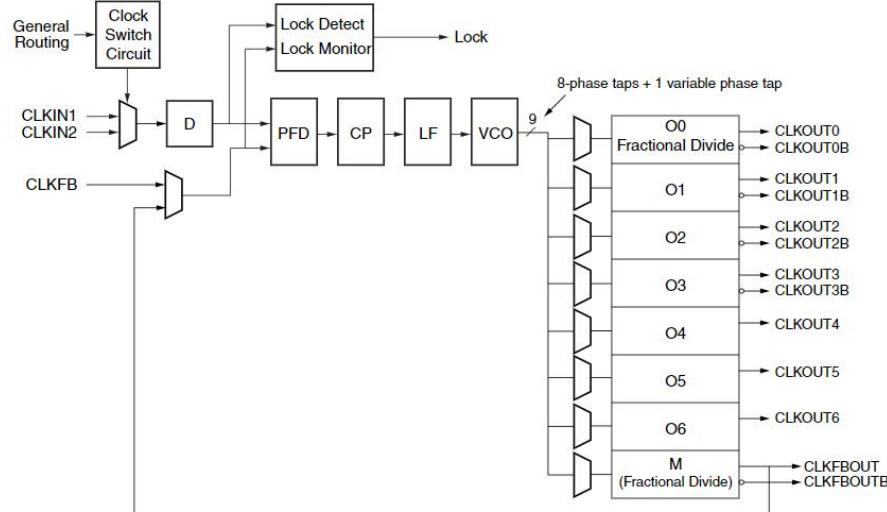
锁相环其实是一个闭环负反馈的相位误差控制系统，环路通过比较差错检波器两个输入信号之间的比较结果，然后通过此结果来调整压控振荡器输出的信号频率，最终达到两个输入信号同频的目的，并且相位误差处于一个稳定的状态。

FPGA概述-架构



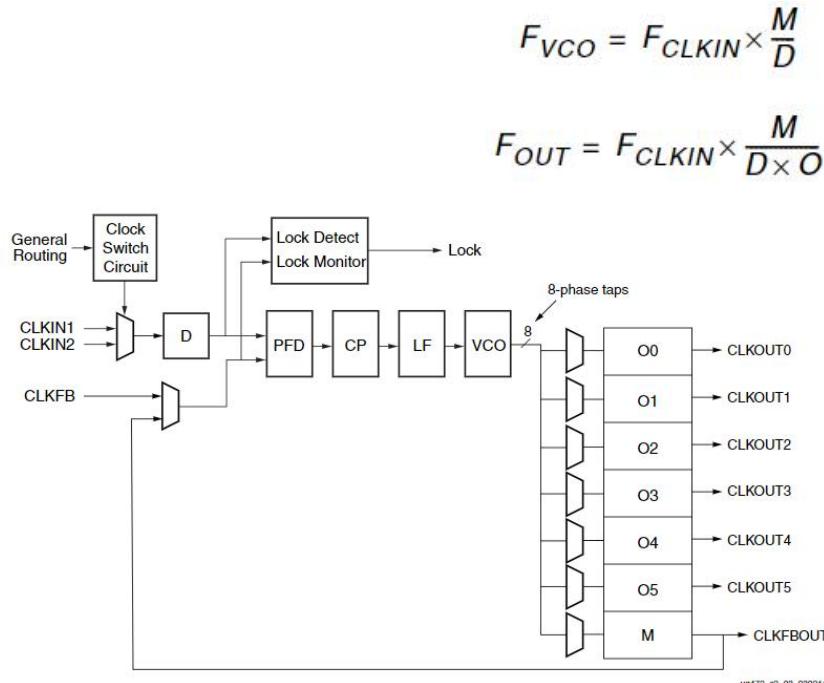
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

CMT



ug472_c2_02_020712

Figure 3-2: Detailed MMCM Block Diagram



ug472_c2_03_030211

Figure 3-3: Detailed PLL Block Diagram

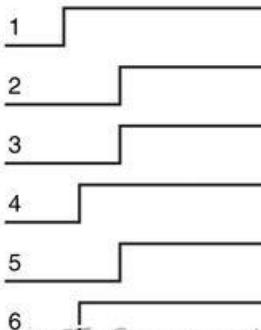
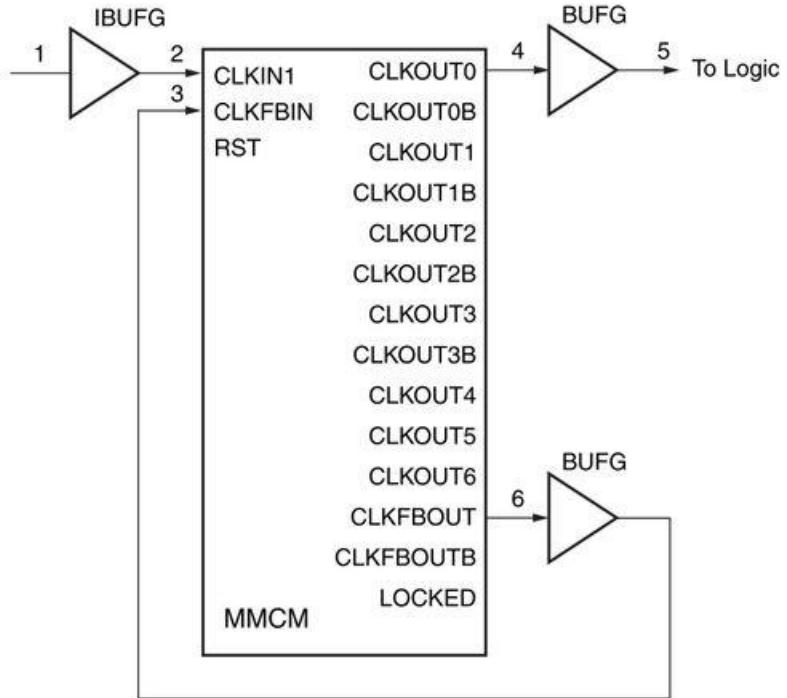
- PLL是简化版的MMCM，结构与MMCM大体一致，但没有反相输出、分数倍数、精确的动态相位调整以及CLKOUT6

FPGA概述-架构



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

CMT



知乎 @sazczrn
UG472_c2_11_061710

FPGA概述-Xilinx产品介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

Xilinx 提供多节点产品系列

45nm	28nm	20nm	16nm
SPARTAN ⁶	VIRTEX ⁷ KINTEX ⁷ ARTIX ⁷ SPARTAN ⁷	VIRTEX ⁷ UltraSCALE KINTEX ⁷ UltraSCALE	VIRTEX ⁷ UltraSCALE+ KINTEX ⁷ UltraSCALE+

◆ SPARTAN6

- [工业网络](#)
- [汽车网络和连接功能](#)
- [高分辨率视频和图形](#)

◆ SPARTAN7

- [机器视觉接口连接](#)
- [单轴马达控制](#)

◆ ARTIX7

- [64 通道便携超声](#)
- [无线回传：点对点 1024QAM 微波调制解调器](#)
- [多协议机器视觉摄像机](#)

◆ Kintex7

- [LED 背光平板显示器和 3DTV](#)
- [LTE 基带解决方案](#)

◆ Virtex7

- [100GE 线卡](#)
- [10GPON/10GEAPON OLT 线路卡](#)

◆ Virtex UltraScale

- [4x100G 转发器](#)
- [400G MAC - Interlaken Bridge](#)
- [ASIC 原型 & 仿真](#)

◆ Kintex UltraScale

- [远程无线电头端 DFE 8x8](#)
- [100MHz TD-LTE 无线电单元](#)
- [256 通道医疗超声波图像处理](#)

◆ Virtex UltraScale+

- [5G基带](#)
- [有线通信](#)
- [雷达](#)
- [测试测量](#)

◆ Kintex UltraScale+

- [PON接入](#)
- [移动回程](#)
- [数据中心网络加速](#)

FPGA概述-Xilinx产品介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Artix-7 FPGAs

Transceiver Optimization at the Lowest Cost and Highest DSP Bandwidth (1.0V, 0.95V, 0.9V)									
	Part Number	XC7A12T	XC7A15T	XC7A25T	XC7A35T	XC7A50T	XC7A75T	XC7A100T	XC7A200T
Logic Resources	Logic Cells	12,800	16,640	23,360	33,280	52,160	75,520	101,440	215,360
	Slices	2,000	2,600	3,650	5,200	8,150	11,800	15,850	33,650
	CLB Flip-Flops	16,000	20,800	29,200	41,600	65,200	94,400	126,800	269,200
Memory Resources	Maximum Distributed RAM (Kb)	171	200	313	400	600	892	1,188	2,888
	Block RAM/FIFO w/ ECC (36 Kb each)	20	25	45	50	75	105	135	365
	Total Block RAM (Kb)	720	900	1,620	1,800	2,700	3,780	4,860	13,140
Clock Resources	CMTs (1 MMCM + 1 PLL)	3	5	3	5	5	6	6	10
I/O Resources	Maximum Single-Ended I/O	150	250	150	250	250	300	300	500
	Maximum Differential I/O Pairs	72	120	72	120	120	144	144	240
	DSP Slices	40	45	80	90	120	180	240	740
Embedded Hard IP Resources	PCIe® Gen ⁽¹⁾	1	1	1	1	1	1	1	1
	Analog Mixed Signal (AMS) / XADC	1	1	1	1	1	1	1	1
	Configuration AES / HMAC Blocks	1	1	1	1	1	1	1	1
Speed Grades	GTP Transceivers (6.6 Gb/s Max Rate) ⁽²⁾	2	4	4	4	4	8	8	16
	Commercial Temp (C)	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2
	Extended Temp (E)	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3
	Industrial Temp (I)	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L
	Package ^{(3), (4)}	Dimensions (mm)	Ball Pitch (mm)						
	CPG236	10 x 10	0.5		106 (2)		106 (2)		
	CPG238	10 x 10	0.5	112 (2)		112 (2)			
	CSG324	15 x 15	0.8		210 (0)		210 (0)		210 (0)
	CSG325	15 x 15	0.8	150 (2)	150 (4)	150 (4)	150 (4)		
	FTG256	17 x 17	1.0		170 (0)		170 (0)		170 (0)
	SBG484	19 x 19	0.8						285 (4)
	FGG484 ⁽⁵⁾	23 x 23	1.0		250 (4)		250 (4)	285 (4)	285 (4)
Footprint Compatible	FBG484 ⁽⁵⁾	23 x 23	1.0						
Footprint Compatible	FGG676 ⁽⁶⁾	27 x 27	1.0				300 (8)	300 (8)	400 (8)
	FBG676 ⁽⁶⁾	27 x 27	1.0						
	FFG1156	35 x 35	1.0						500 (16)

Available User I/O: 3.3V SelectIO™ HR I/O (GTP Transceivers)



THANKS



國科大杭州高等研究院
Hangzhou Institute for Advanced Study, UCAS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计

第二讲 (2.1 Vivado工具使用&Lab0.1)

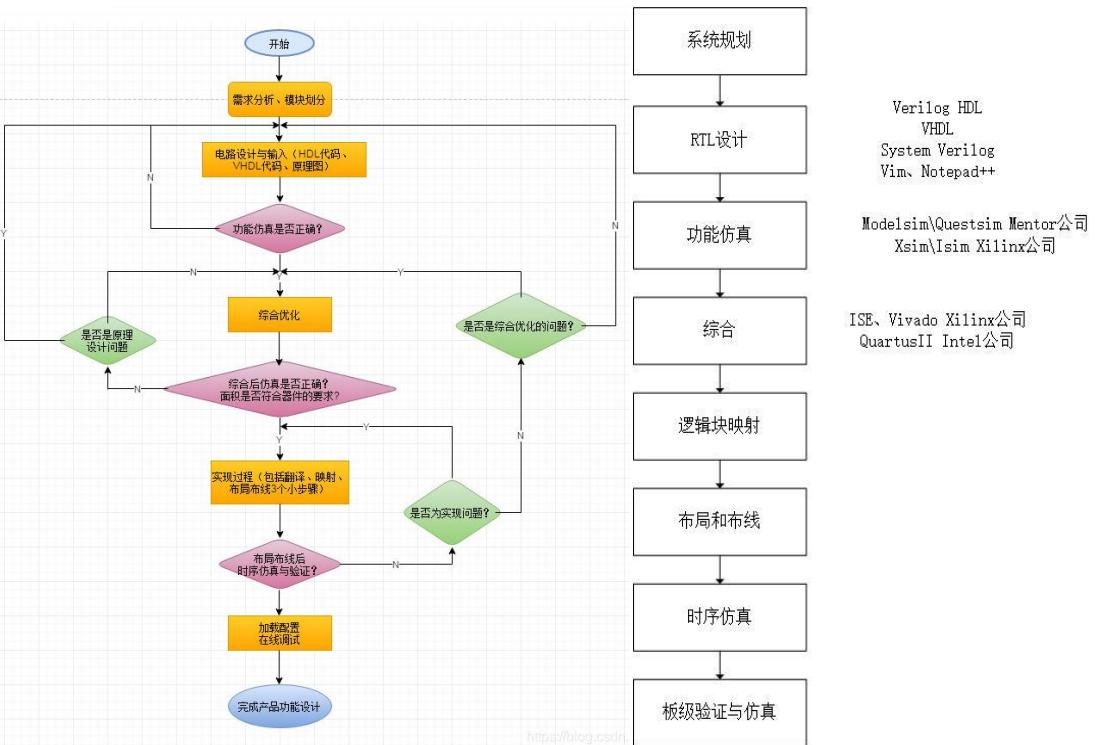
2025年3月11日

软件工具——Vivado设计流程



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

现代EDA的开发流程：



使用Vivado软件快速开发流程：

- 通过原理图或编写HDL文件的方式创建设计
- 功能仿真(可选步骤)
- 综合 (类似于电路设计的原理图阶段)
- 通过I/O Planning添加管脚约束或通过编写约束文件添加管脚约束
- 添加时序约束(可选步骤)
- 布局布线(类似于电路设计的PCB设计阶段)
- 时序仿真(可选步骤)
- 生成Bitstream文件
- 将生成的Bitstream文件下载到FPGA芯片



HDL语言为硬件描述语言 (Hardware Description Language)

- 与高级程序语言 (C) 类似, Verilog HDL是一种高级程序设计语言
- C语言与Verilog HDL设计思想完全不同
- 软件高级程序语句是对通用型处理器 (如CPU) 的编程, 主要是在固定硬件体系结构下的软件化程序设计。处理器的体系结构和功能决定了可以用于程序编程的**固定指令集**, 设计人员的工作是调用这些指令, 在固化的体系结构下实现特定的功能
- Verilog HDL和VHDL等**硬件描述语言**是对电路的设计, 将基本的最小数字电路单元 (如门单元、寄存器、存储器等) 通过**连接方式**构成具有特定功能的**硬件电路**
- HDL程序设计的**正确性**需要通过综合后电路的**正确性**来验证, 逻辑上相同, 物理电路的形式有可能完全不同

体会一下HDL语言与C语言的不同

Verilog HDL设计8位计数器

```
51 module counter(clk,rst_n,cnt
52   input clk,rst_n;
53   output[7:0] cnt;
54 );
55   reg [7:0] cnt;
56   always @(posedge clk or negedge rst_n)
57     if (!rst_n)cnt<=8'h00000000;
58     else cnt<=cnt+1'b1;
59 endmodule
```

下面代码套用了C语言的思路

```
60 module counter(clk,rst_n,cnt
61   input clk,rst_n;
62   output[7:0] cnt;
63 );
64   reg [7:0] cnt;
65   integer i;
66   always @(posedge clk or negedge rst_n) begin
67     if (!rst_n)cnt<=8'h00000000;
68     else
69       for (i = 0 ;i<=255 ;i=i+1 ) begin
70         cnt<=cnt+1'b1;
71       end
72 endmodule
```

可综合的Verilog HDL语法



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

◆ 可综合语法的特点

- ◆ 是指硬件能够实现的语法
- ◆ 可综合语法的子集很小

◆ 模块声明语法： module...endmodule

- ◆ 在每个Verilog文件中都会出现该语法，它是一个固定的用法
- ◆ Module后的name为该module的命名，取名没有任何限制（默认数字、下划线和字母的组合均可）；随后一个“()”内罗列出该模块所有的输入输出端口信号名；接着一直到endmodule之间就是要实现功能的逻辑代码

```
module name(  
    端口信号1,  
    端口信号2,  
    端口信号3,  
    ...  
    端口信号N  
);  
  
// 逻辑代码  
// 逻辑代码  
  
endmodule
```

```
module Lab01_hdl(  
    input and_SW0,  
    input and_SW1,  
    output and_LD1_0  
);  
  
assign and_LD1_0 = and_SW0 & and_SW1;  
  
endmodule
```

可综合的Verilog HDL语法



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

◆ 端口声明：input, output, inout

- ◆ inout用法比较特殊，不常用
- ◆ 每个module都会有输入输出的信号用于和外部器件或其他module进行接口
- ◆ 对于本地module而言，这些信号无非可以归为三类，即输入（input）信号，输出（output）信号和双向（inout）信号
- ◆ 这些端口信号名都要在module名后的“（）”内列出

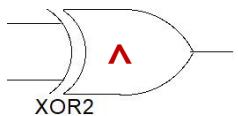
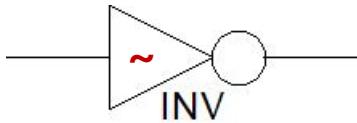
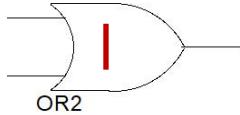
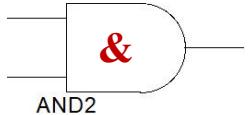
```
22 module name(  
23   input clk,  
24   input rst,  
25   output [7:0] led);  
26   /*<逻辑代码>...*/  
27 endmodule
```

可综合的Verilog HDL语法



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

◆按位逻辑运算符: ~, &, |, ^, ~^或^~



可综合的Verilog HDL语法



◆连续赋值：assign

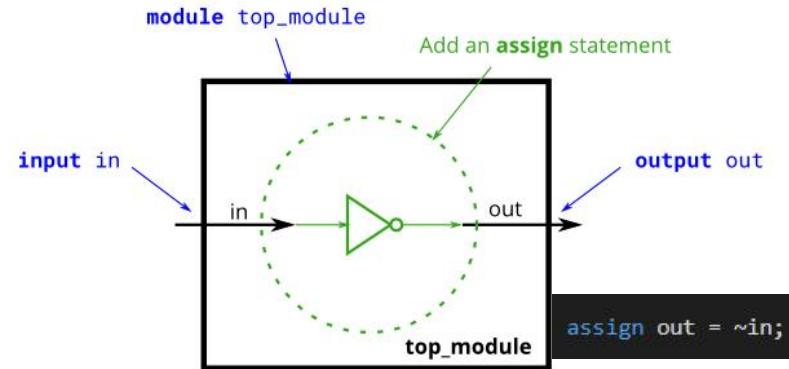
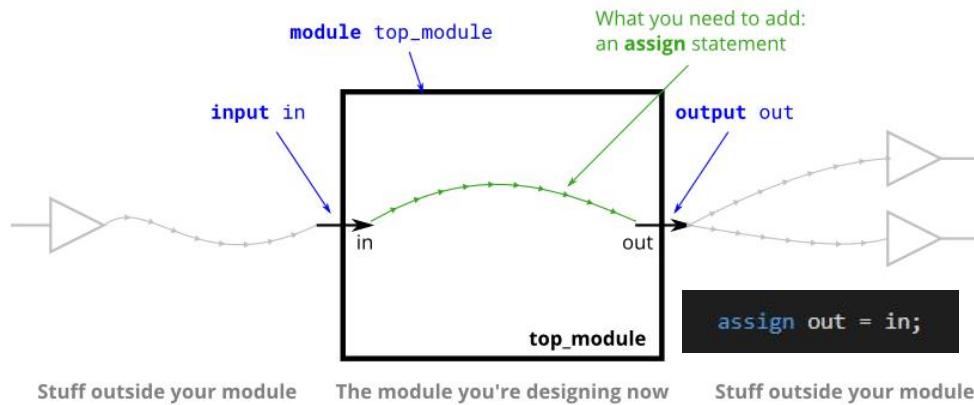
assign相当于连线，电路行为上是将一个变量的值不间断地赋值给另一个变量，比如：把一个模块的输出给另一个模块当输入。

assign的功能属于组合逻辑的范畴，其表达的功能：

- (1) 持续赋值；（从电路行为层面）
- (2) 连线；（从实际硬件电路层面）

```
assign<wire变量名> = <赋值表达式>;
```

```
assign a = b;  
assign and_LD1_0 = and_SW0 & and_SW1;
```



实验0.1：基本门电路的硬件实现

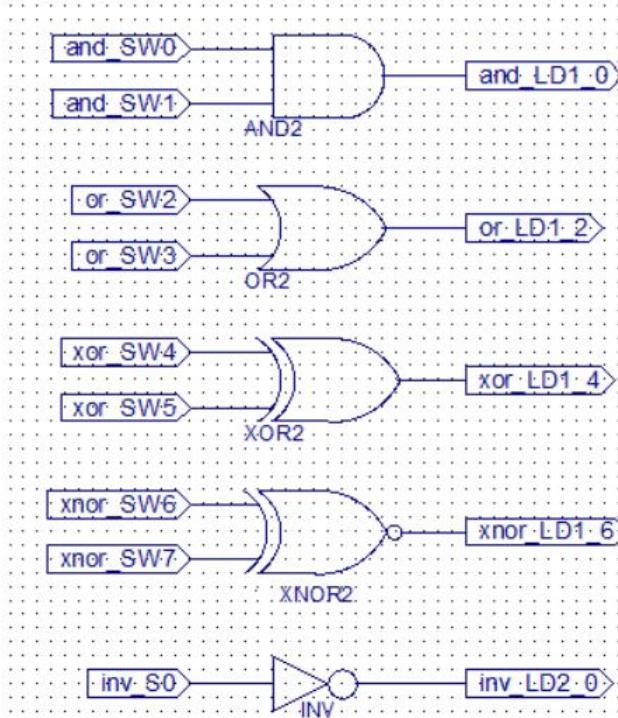


■ 实验内容

- 基本门电路实验，包括与门、或门、异或门、同或门、非门
- 使用DIP8、按键作为输入，使用LED灯作为输出
- 基于Vivado文本输入方式进行设计
- 熟悉Vivado设计工具的基本使用

■ 电路表达式

- $\text{and_LD1_0} = \text{and_SW0} \& \text{and_SW1}$
- $\text{or_LD1_2} = \text{or_SW2} \mid \text{or_SW3}$
- $\text{xor_LD1_4} = \text{xor_SW4} \wedge \text{xor_SW5}$
- $\text{xnor_LD1_6} = \text{xnor_SW6} \sim \text{xnor_SW7}$
- $\text{inv_LD2_0} = \sim \text{inv_S0}$



电路原理图

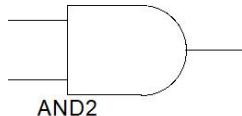
实验0.1：基本门电路的硬件实现



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

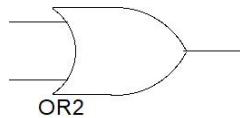
x	0	0	1	1
y	0	1	0	1
z	0	0	0	1

与门



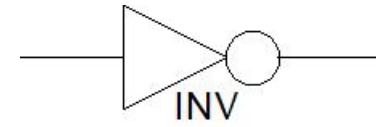
x	0	0	1	1
y	0	1	0	1
z	0	1	1	1

或门



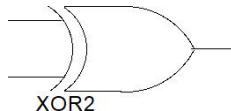
x	0	1
z	1	0

非门



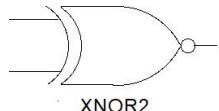
x	0	0	1	1
y	0	1	0	1
z	0	1	1	0

异或门



x	0	0	1	1
y	0	1	0	1
z	1	0	0	1

同或门



实验0.1：基本门电路的硬件实现



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

序号	信号名称	输入/输出	信号描述	FPGA引脚	说明
1	and_SW0	输入	与门输入1	T5	ON为高电平
2	and_SW1	输入	与门输入2	T3	ON为高电平
3	or_SW2	输入	或门输入1	R3	ON为高电平
4	or_SW3	输入	或门输入2	V4	ON为高电平
5	xor_SW4	输入	异或门输入1	V5	ON为高电平
6	xor_SW5	输入	异或门输入2	V2	ON为高电平
7	xnor_SW6	输入	同或门输入1	U2	ON为高电平
8	xnor_SW7	输入	同或门输入2	U3	ON为高电平
9	inv_SO	输入	非门输入	R11	按下为高电平
10	and_LD1_0	输出	与门输出	K3	高电平点亮
11	or_LD1_2	输出	或门输出	L1	高电平点亮
12	xor_LD1_4	输出	异或门输出	J5	高电平点亮
13	xnor_LD1_6	输出	同或门输出	H6	高电平点亮
14	inv_LD2_0	输出	非门输出	K2	高电平点亮

输入输出端口表

实验0.1：基本门电路的硬件实现



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

code.visualstudio.com

HIT-NAS HOME-NAS FPGA Lab-NAS 统一身份认证平台 光电成像技术 杭州政府

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn Search Docs Download

Version 1.87 is now available! Read about the new features and fixes from February.

Code editing. Redefined.

Free. Built on open source. Runs everywhere.

Download for Windows Stable Build

Web, Insiders edition, or other platforms

By using VS Code, you agree to its license and privacy statement.

建议使用Visual Studio Code软件进行设计开发

The screenshot shows the official Visual Studio Code website. At the top, there's a navigation bar with links to 'Docs', 'Updates', 'Blog', 'API', 'Extensions', 'FAQ', and 'Learn'. Below the navigation is a message about Version 1.87. The main area features a large heading 'Code editing. Redefined.' with the subtext 'Free. Built on open source. Runs everywhere.'. Below this are download buttons for 'Windows' (Stable Build) and 'Web, Insiders edition, or other platforms'. A note at the bottom states 'By using VS Code, you agree to its license and privacy statement.' To the right, a screenshot of the Visual Studio Code interface is shown, displaying a code editor with several tabs (App.js, index.js, serviceWorker.js) and an extensions marketplace sidebar listing various tools like Python, GitLens, C/C++, ESLint, Debugger for Chrome, Language Support, vscode-icons, and Vetur.

实验0.1：基本门电路的硬件实现



The screenshot shows the Vivado 2019.1 software interface. The 'Tools' menu is highlighted with a red box. A sub-menu 'Settings' is open, also highlighted with a red box. The 'Text Editor' section is selected and highlighted with a red box. The 'Current Editor' dropdown shows 'Custom Editor...' with a red box around it. A 'Custom Editor Definition' dialog box is open, showing the path 'C:/Users/mzpan/AppData/Local/Programs/Microsoft VS Code/Code.exe [file name]' in the 'Editor' field, which is also highlighted with a red box. Below the dialog are two red text annotations:

- 1.注意VS Code路径以及斜杠方向
- 2.完整路径后加上空格[file name]

实验0.1：基本门电路的硬件实现



File Edit Selection View Go Run Terminal Help

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. [Manage](#) [Learn More](#)

Lab1_led.v X 23

C: > Xilinx > MyProject_2022 > Lab1_led > Lab1_led.srcs > sources_1 > new > Lab1_led.v

1 //timescale 1ns / 1ps
2 ///////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 2022/03/07 11:22:54
7 // Design Name:
8 // Module Name: Lab1_led
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////
21
22
23 module Lab1_led(
24 input key_all,
25 input key_a,
26 input key_b,
27 output led
28);
29 endmodule

解除VS Code的限制模式

实验0.1：基本门电路的硬件实现



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

The screenshot shows the Visual Studio Code interface with a dark theme. At the top, there's a menu bar with File, Edit, Selection, View, Go, Run, Terminal, Help. Below the menu is a toolbar with icons for file operations like Open, Save, and Find. A status bar at the bottom indicates "Workspace Trust - Visual Studio Code". A message bar says "Restricted Mode is intended for safe code browsing. Trust this window to enable all features." with links to Manage and Learn More.

The main area displays two sections: "In a Trusted Window" and "In Restricted Mode".

- In a Trusted Window:** You trust the authors of the files in the current window. All features are enabled:
 - ✓ Tasks are allowed to run
 - ✓ Debugging is enabled
 - ✓ All extensions are enabledA blue "Trust" button is centered below this section.
- In Restricted Mode:** You do not trust the authors of the files in the current window. The following features are disabled:
 - ✗ Tasks are not allowed to run
 - ✗ Debugging is disabled
 - ✗ 7 extensions are disabled or have limited functionality

At the bottom, under "Trusted Folders & Workspaces", it says "You trust the following folders, their subfolders, and workspace files." It lists a single entry: "Host: Local, Path: c:\Xilinx\MyProject_2022". A red box highlights the "Add Folder" button.

**1.选择信任当前窗口
2.建议将工程目录加入信任文件夹**

实验0.1：基本门电路的硬件实现



The screenshot shows the Visual Studio Code Marketplace interface. On the left, there is a sidebar with various icons and a search bar at the top. Below the search bar, the text "Restricted Mode is intended for safe code browsing. Trust this window to enable all features." is displayed, along with "Manage" and "Learn More" buttons. The main area is titled "EXTENSIONS: MARKETPLACE". A search bar at the top of this section has "verilog" typed into it. A red box highlights this search term. Below the search bar, a list of extensions is shown:

- verilog** (highlighted with a red box): An extension aim at making... Gtylcaro&Gewinn. Status: 76K. Buttons: Install.
- Verilog-HDL/SystemVerilog**: An extension aim at making... Masahiro Hiramori. Status: 759,656. Buttons: Uninstall, Settings (highlighted with a red box), Learn More.
- Verilog HDL**: Verilog HDL Language Supp... leafmaple. Status: 121K. Buttons: Install.
- Verilog_Testben...**: verilog-testbench-instance Truecrab. Status: 64K. Buttons: Install.
- Verilog Format**: Console application for appl... Ericson Joseph. Status: 59K. Buttons: Install.
- Verilog Snippet**: A snippet for verilog czh. Status: 31K. Buttons: Install.
- verilog-format...**: A Verilog code formatter us... IsaacT. Status: 28K. Buttons: Install.
- Verilog Highlight**: Syntax highlighter for Verilog tzylee. Status: 15K. Buttons: Install.
- verilog-utils**: Buttons: Install.

On the right side of the main content area, there is a detailed view of the "Verilog-HDL/SystemVerilog/Bluespec SystemVerilog" extension page. This page includes:

- A large image of an FPGA chip.
- The extension name: Verilog-HDL/SystemVerilog/Bluespec SystemVerilog.
- The developer: Masahiro Hiramori.
- The download count: 759,656.
- The rating: ★★★★★ (20).
- A status message: "This extension has been disabled because the current workspace is not trusted." (highlighted with a red box).
- Buttons: Uninstall, Settings (highlighted with a red box), Learn More.
- A "HDL support for VS Code" section with a description, installs (760k), downloads (4.1M), and a preview of the code editor.
- Categories: Programming Languages, Snippets, Linters, Formatters.
- Resources: Marketplace, Issues, Repository, License, Masahiro Hiramori.
- More Info: Published 2015-12-05, 03:04:48.

实验0.1：基本门电路的硬件实现



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

The screenshot shows the Microsoft Visual Studio Code interface with the Extensions Marketplace open. The search bar at the top right contains the text "Extension: Verilog-HDL/SystemVerilog/Bluespec SystemVerilog". The results list includes several extensions related to Verilog and SystemVerilog, such as "verilog", "Verilog HDL", "Verilog Testben...", "Verilog Format", "Verilog Snippet", "verilog-format...", "Verilog Highlight", "verilog-utils", "SystemVerilog ...", "verilog-autoline", "vscode-verilog", and "Verilog Testben...". The "verilog" extension is currently selected, as indicated by the highlighted status bar message "Extension: verilog". On the right side of the screen, the "Settings" sidebar is open, showing configuration options for the "Verilog" extension. A red box highlights the dropdown menu under "Verilog > Linting: Linter". The dropdown menu lists several possible values: "none", "xvlog" (which is selected), "iverilog", "verilator", "modelsim", "slang", and "none". The "xvlog" option is highlighted with a red box. Below the dropdown, there is a note: "Select the Verilog linter. Possible values are 'iverilog', 'verilator', 'modelsim', 'xvlog', 'slang' or 'none'." At the bottom of the settings sidebar, there is another dropdown menu for "Verilog > Linting: Modelsim: Run At File Location".

实验0.1：基本门电路的硬件实现



The screenshot shows the Vivado 2019.1 software interface. At the top, there is a menu bar with File, Flow, Tools, Window, Help, and a Quick Access search bar. The main window has three main sections:

- Quick Start:** Contains "Create Project >" (highlighted with a red box), "Open Project >", and "Open Example Project >". There is also a red button labeled "建立新工程" (Create New Project).
- Tasks:** Contains "Manage IP >", "Open Hardware Manager >", and "Xilinx Tcl Store >".
- Learning Center:** Contains "Documentation and Tutorials >", "Quick Take Videos >", and "Release Notes Guide >".

At the bottom, there is a toolbar with icons for File, Search, Home, Project, IP, Flow, Tools, Window, Help, and a Tcl Console tab. A status bar at the bottom right shows the time as 18:33 and the date as 2021/3/13.

实验0.1：基本门电路的硬件实现



The screenshot shows the Vivado 2019.1 software interface. On the left, there's a 'Quick Start' panel with options like 'Create Project >', 'Open Project >', and 'Open Example Project >'. Below it is a 'Tasks' panel with 'Manage IP >', 'Open Hardware Manager >', and 'Xilinx Tcl Store >'. At the bottom is a 'Learning Center' panel with 'Documentation and Tutorials >', 'Quick Take Videos >', and 'Release Notes Guide >'. In the center, a 'New Project' dialog box is open with the title 'Create a New Vivado Project'. It contains instructions: 'This wizard will guide you through the creation of a new project. To create a Vivado project you will need to provide a name and a location for your project files. Next, you will specify the type of flow you'll be working with. Finally, you will specify your project sources and choose a default part.' At the bottom of the dialog are buttons: '< Back', 'Next >' (which is highlighted with a red box), 'Finish', and 'Cancel'. The Xilinx logo is visible at the bottom of the dialog and in the background of the main interface. The Windows taskbar at the bottom shows various pinned icons and the date/time '18:35 2021/3/13'.

实验0.1：基本门电路的硬件实现



File Flow Tools Window Help Q- Quick Access

VIVADO[®]
HLx Editions

Quick Start

Create Project >
Open Project >
Open Example Project >

Tasks

Manage IP >
Open Hardware Manager >
Xilinx Tcl Store >

Learning Center

Documentation and Tutorials >
Quick Take Videos >
Release Notes Guide >

Tcl Console

工程向导 会引导您完成设计源文件和目标器件的选择

New Project

Project Name
Enter a name for your project and specify a directory where the project data files will be stored.

Project name: Lab0_hdl 工程名称

Project location: C:/Xilinx/MyProject/2024 工程路径

Create project subdirectory

Project will be created at: C:/Xilinx/MyProject/2024/Lab0_HDL

? < Back Next > Finish Cancel

XILINX[®]

实验0.1：基本门电路的硬件实现



The screenshot shows the Vivado 2019.1 interface. On the left, there's a 'Quick Start' panel with options like 'Create Project >', 'Open Project >', and 'Open Example Project >'. Below it is a 'Tasks' panel with 'Manage IP >', 'Open Hardware Manager >', and 'Xilinx Tcl Store >'. At the bottom is a 'Learning Center' panel with 'Documentation and Tutorials >', 'Quick Take Videos >', and 'Release Notes Guide >'. A central window titled 'New Project' is open, showing the 'Project Type' selection step. It lists five options: 'RTL Project' (selected), 'Post-synthesis Project', 'JIO Planning Project', 'Imported Project', and 'Example Project'. Each option has a detailed description below it. The 'Next >' button at the bottom right of the dialog is highlighted with a red border. The Xilinx logo is visible in the top right corner of the main window.

实验0.1：基本门电路的硬件实现



The screenshot shows the Vivado 2017.4 software interface. On the left, there is a 'Quick Start' panel with options like 'Create Project >', 'Open Project >', and 'Open Example Project >'. Below it is a 'Tasks' panel with 'Manage IP >', 'Open Hardware Manager >', and 'Xilinx Tcl Store >'. At the bottom is a 'Learning Center' panel with links to 'Documentation and Tutorials >', 'Quick Take Videos >', and 'Release Notes Guide >'. The main area features a 'New Project' wizard titled 'Add Sources'. It contains a list area with a '+' button, a 'Create File' button highlighted with a red box, and checkboxes for 'Scan and add RTL include files into project', 'Copy sources into project', and 'Add sources from subdirectories' (which is checked). Below these are dropdowns for 'Target language: Verilog' and 'Simulator language: Mixed', both also highlighted with red boxes. At the bottom of the wizard are buttons for '?', '< Back' (disabled), 'Next >', 'Finish' (disabled), and 'Cancel'.

XILINX
ALL PROGRAMMABLE.

实验0.1：基本门电路的硬件实现



File Flow Tools Window Help Q Quick Access

VIVADO HLx Editions

Quick Start

- Create Project >
- Open Project >
- Open Example Project >

Tasks

- Manage IP >
- Open Hardware Manager >
- Xilinx Tcl Store >

Learning Center

- Documentation and Tutorials >
- Quick Take Videos >
- Release Notes Guide >

New Project

Add Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

Create Source File

Create a new source file and add it to your project.

File type: Verilog

File name: Lab0_hdl 输入HDL源文件名称

File location: Local to Project

OK Cancel

Scan and add RTL include files into project

Copy sources into project

Add sources from subdirectories

Target language: Verilog Simulator language: Mixed

? Back Next > Finish Cancel

XILINX

实验0.1：基本门电路的硬件实现



VIVADO
HLx Editions

Quick Start

- Create Project >
- Open Project >
- Open Example Project >

Tasks

- Manage IP >
- Open Hardware Manager >
- Xilinx Tcl Store >

Learning Center

- Documentation and Tutorials >
- Quick Take Videos >
- Release Notes Guide >

New Project

Add Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

Index	Name	Library	HDL Source For	Location
1	Lab0_hdl.v	xil_defaultlib	Synthesis & Simulation	<Local to Project>

Add Files Add Directories Create File

Scan and add RTL include files into project
 Copy sources into project
 Add sources from subdirectories

Target language: Verilog Simulator language: Mixed

? < Back Next > Finish Cancel

XILINX

实验0.1：基本门电路的硬件实现



The screenshot shows the Vivado 2019.1 software interface. On the left, there's a sidebar with sections for 'Quick Start', 'Tasks', and 'Learning Center'. The 'Quick Start' section has links to 'Create Project >', 'Open Project >', and 'Open Example Project >'. The 'Tasks' section has links to 'Manage IP >', 'Open Hardware Manager >', and 'Xilinx Tcl Store >'. The 'Learning Center' section has links to 'Documentation and Tutorials >', 'Quick Take Videos >', and 'Release Notes Guide >'. In the center, a 'New Project' dialog box is open. It has a title 'Add Constraints (optional)' with the sub-instruction 'Specify or create constraint files for physical and timing constraints.' Below this is a large empty area with a toolbar at the top containing buttons for adding (+), removing (-), and reordering files. A note says 'Use Add Files or Create File buttons below'. At the bottom of this area are 'Add Files' and 'Create File' buttons, with 'Create File' being highlighted with a red box. To its right, the text '创建管脚约束文件' (Create Pin Constraint File) is displayed in red. At the very bottom of the dialog are buttons for '?', '< Back' (disabled), 'Next >', 'Finish' (disabled), and 'Cancel'. On the right side of the interface, there's a vertical panel with project names: 'ct/2024/Lab0_sch' and 'ct/2021/Lab0_sch'. The XILINX logo is visible in the top right corner of the interface.

实验0.1：基本门电路的硬件实现



File Flow Tools Window Help Q: Quick Access

VIVADO HLx Editions

Quick Start

- Create Project >
- Open Project >
- Open Example Project >

Tasks

- Manage IP >
- Open Hardware Manager >
- Xilinx Tcl Store >

Learning Center

- Documentation and Tutorials >
- Quick Take Videos >
- Release Notes Guide >

New Project

Add Constraints (optional)
Specify or create constraint files for physical and timing constraints.

Create Constraints File

Create a new constraints file and add it to your project

File type: XDC

File name: Lab0_hdl

File location: <Local to Project>

OK Cancel

输入管脚约束文件名称

Add Files Create File

Copy constraints files into project

< Back Next > Finish Cancel

XILINX

实验0.1：基本门电路的硬件实现



The screenshot shows the Vivado HLx Editions interface. On the left, there's a 'Quick Start' sidebar with options like 'Create Project >', 'Open Project >', and 'Open Example Project >'. Below it is a 'Tasks' sidebar with 'Manage IP >', 'Open Hardware Manager >', and 'Xilinx Tcl Store >'. At the bottom is a 'Learning Center' sidebar with 'Documentation and Tutorials >', 'Quick Take Videos >', and 'Release Notes Guide >'. A central window titled 'New Project' is open, prompting for 'Add Constraints (optional)'. It shows a list with 'Lab0_hdl.xdc' selected and '

XILINX®

实验0.1：基本门电路的硬件实现



Vivado 2019.1

File Flow Tools Window Help Q: Quick Access

VIVADO HLx Editions

Quick Start

Create Project >
Open Project >
Open Example Project >

Tasks

Manage IP >
Open Hardware Manager >
Xilinx Tcl Store >

Learning Center

Documentation and Tutorials >
Quick Take Videos >
Release Notes Guide >

New Project

Default Part

Choose a default Xilinx part or board for your project.

选择FPGA器件

Parts | Boards

Reset All Filters

Category: All Package: csg324 Temperature: All Remaining

Family: Artix-7 Speed: -1 Static power: All Remaining

Search: Q-

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Tran
xc7a15tcsg324-1	324	210	10400	20800	25	0	45	0
xc7a35tcsg324-1	324	210	20800	41600	50	0	90	0
xc7a50tcsg324-1	324	210	32600	65200	75	0	120	0
xc7a75tcsg324-1	324	210	47200	94400	105	0	180	0
xc7a100tcsg324-1	324	210	63400	126800	135	0	240	0

< Back Next > Finish Cancel

Tcl Console

工程向导 会引导你完成设计源文件和目标器件的选择

Windows Search Task View Start File Explorer Twitter Word Excel

18:44 2021/3/13

XILINX

实验0.1：基本门电路的硬件实现



File Flow Tools Window Help Q- Quick Access

VIVADO
HLx Editions

Quick Start

Create Project >
Open Project >
Open Example Project >

Tasks

Manage IP >
Open Hardware Manager >
Xilinx Tcl Store >

Learning Center

Documentation and Tutorials >
Quick Take Videos >
Release Notes Guide >

XILINX

New Project

New Project Summary

- 1 A new RTL project named 'Lab0_hdl' will be created.
- 1 source file will be added.
- 1 constraints file will be added.
- The default part and product family for the new project:
Default Part: xc7a35tcsg324-1
Product: Artix-7
Family: Artix-7
Package: csg324
Speed Grade: -1

To create the project, click Finish

< Back Next > **Finish** Cancel

XILINX

实验0.1：基本门电路的硬件实现



The screenshot shows the Vivado IDE interface with the 'PROJECT MANAGER - Lab0_hdl' window open. A modal dialog box titled 'Define Module' is in the foreground, prompting the user to define a module and specify I/O Ports. The 'Module Definition' section shows the 'Module name:' field set to 'Lab0_hdl', which is highlighted with a red box and overlaid with the text '1. 输入HDL中模块名称'. Below it is the 'I/O Port Definitions' table, which currently contains one row: 'input' under 'Port Name', 'input' under 'Direction', and '0' under 'MSB'. The 'Implementation' tab of the project summary is visible in the background.

Module name: Lab0_hdl

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
input	input	<input type="checkbox"/>	0	0

Implementation

Status	Not started
messages:	No errors or warnings
part:	xc7a35tcsg324-1
strategy:	Vivado Implementation Defaults
Report Strategy:	Vivado Implementation Default Reports
Incremental implementation:	None

实验0.1：基本门电路的硬件实现



2. 定义模块中所有的端口
(注意输入输出方向)

Port Name	Direction	Bus	MSB	LSB
and_SW0	input	<input type="checkbox"/>		
and_SW1	input	<input type="checkbox"/>		
inv_S0	input	<input type="checkbox"/>		
and_LD1_0	output	<input type="checkbox"/>	0	0
inv_LD2_0	output	<input type="checkbox"/>	0	0

OK Cancel

实验0.1：基本门电路的硬件实现



打开HDL设计文件

PROJECT MANAGER - Lab0_hdl

Sources

Design Sources (1)

Lab0_hdl (Lab0_hdl.v)

Constraints (1)

Simulation Sources (1)

Utility Sources

Properties

Select an object to see properties

Project Summary

Overview | Dashboard

Project name: Lab0_hdl

Project location: C:/Xilinx/MyProject/2024/Lab0_hdl

Product family: Artix-7

Project part: xc7a35tcsg324-1

Top module name: Lab0_hdl

Target language: Verilog

Simulator language: Mixed

Synthesis

Status: Not started

Messages: No errors or warnings

Part: xc7a35tcsg324-1

Strategy: Vivado Synthesis Defaults

Report Strategy: Vivado Synthesis Default Reports

Incremental synthesis: None

Implementation

Status: Not started

Messages: No errors or warnings

Part: xc7a35tcsg324-1

Strategy: Vivado Implementation Defaults

Report Strategy: Vivado Implementation Default Reports

Incremental implementation: None

Tcl Console | Messages | Log | Reports | Design Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMS	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synth)
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado)

实验0.1：基本门电路的硬件实现



The screenshot shows the Xilinx Vivado IDE interface. On the left, the Project Manager displays a list of sources: Lab0_hdl (Lab0_hdl.v). The Source File Properties panel shows the file Lab0_hdl.v is enabled and is a Verilog file located in C:/Xilinx/MyProject/2024/Lab0_hdl/Lab0_hdl.srcs. The Design Runs panel shows two entries: synth_1 and impl_1, both in the 'Not started' state.

The main window displays the Verilog code for the module Lab0_hdl:

```
1 `timescale 1ns / 1ps
2 // Company: |
3 // Engineer: |
4 // Create Date: 2024/03/12 09:09:16
5 // Design Name: |
6 // Module Name: Lab0_hdl
7 // Project Name: |
8 // Target Devices: |
9 // Tool Versions: |
10 // Description: |
11 // Dependencies: |
12 // Revision: |
13 // Revision 0.01 - File Created
14 // Additional Comments: |
15 //
16
17 module Lab0_hdl(
18     input and_SW0,
19     input and_SW1,
20     input inv_S0,
21     output and_LD1_0,
22     output inv_LD2_0
23 );
24 endmodule
```

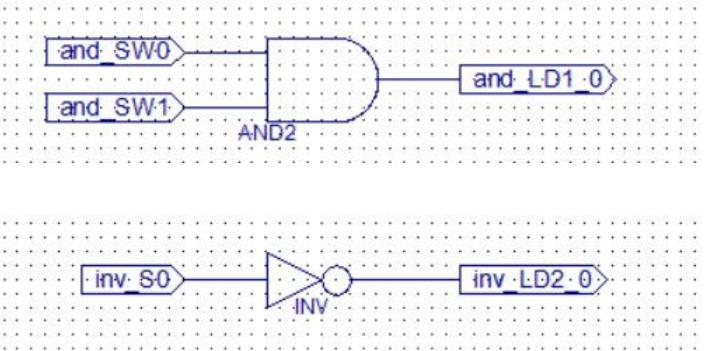
A red box highlights the module definition from line 17 to line 31.

1. 预先定义好的模块

实验0.1：基本门电路的硬件实现



与门，非门电路源代码示例



2. 完成hdI设计输入

3. 保存hdI设计文件

```
>Welcome Lab0_hdl.v
C: > Xilinx > MyProject > 2024 > Lab0_hdl > Lab0_hdl.srccs > sources_1 > new > Lab0_hdl.v
1 `timescale 1ns / 1ps
2 ///////////////////////////////////////////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 2024/03/12 09:09:16
7 // Design Name:
8 // Module Name: Lab0_hdl
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22
23 module Lab0_hdl(
24     input and_SW0,
25     input and_SW1,
26     input inv_S0,
27     output and_LD1_0,
28     output inv_LD2_0
29 );
30
31     assign and_LD1_0 = and_SW0 & and_SW1;
32     assign inv_LD2_0 = ~ inv_S0;
33
34 endmodule
35
```

实验0.1：基本门电路的硬件实现



The screenshot shows the Vivado IDE interface. On the left, the Project Manager sidebar lists various project categories: PROJECT MANAGER, IP INTEGRATOR, SIMULATION, RTL ANALYSIS, SYNTHESIS, IMPLEMENTATION, and PROGRAM AND DEBUG. The PROJECT MANAGER section is currently active, displaying the Sources tree. A red box highlights the 'Lab0_hdl.xdc' file under Constraints. The main workspace contains three main panels: 1) PROJECT SUMMARY (Overview and Dashboard), 2) DESIGN RUNS (Synthesis and Implementation status), and 3) SOURCE FILE PROPERTIES (for Lab0_hdl.xdc). The DESIGN RUNS panel shows two runs: synth_1 and impl_1, both in the 'Not started' state.

打开管脚约束文件

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synth)
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado

实验0.1：基本门电路的硬件实现



```
>Welcome      Lab0_hdl.v      Lab0_hdl.xdc      EGo1.xdc •  
C: > Xilinx > MyProject > 2024 > EGo1.xdc  
35 ///////////////////////////////////////////////////////////////////5个按键/////////////////////////////////////////////////////////////////  
  
inv_S0      set_property -dict {PACKAGE_PIN R11 IOSTANDARD LVCMOS33} [get_ports {btn_pin[0]}]  
37 set_property -dict {PACKAGE_PIN R17 IOSTANDARD LVCMOS33} [get_ports {btn_pin[1]}]  
38 set_property -dict {PACKAGE_PIN R15 IOSTANDARD LVCMOS33} [get_ports {btn_pin[2]}]  
39 set_property -dict {PACKAGE_PIN V1 IOSTANDARD LVCMOS33} [get_ports {btn_pin[3]}]  
40 set_property -dict {PACKAGE_PIN U4 IOSTANDARD LVCMOS33} [get_ports {btn_pin[4]}]  
41
```

and_SW0
and_SW1

inv_LD2_0

and_LD1_0

```
>Welcome      Lab0_hdl.v      Lab0_hdl.xdc      EGo1.xdc •  
C: > Xilinx > MyProject > 2024 > EGo1.xdc  
48 set_property -dict {PACKAGE_PIN M4 IOSTANDARD LVCMOS33} [get_ports {sw_pin[5]}]  
49 set_property -dict {PACKAGE_PIN N4 IOSTANDARD LVCMOS33} [get_ports {sw_pin[6]}]  
50 set_property -dict {PACKAGE_PIN R1 IOSTANDARD LVCMOS33} [get_ports {sw_pin[7]}]  
51  
/////////////////////////////////////////////////////////////////拨码开关sw8~sw15/////////////////////////////////////////////////////////////////  
52 set_property -dict {PACKAGE_PIN U3 IOSTANDARD LVCMOS33} [get_ports {dip_pin[0]}]  
53 set_property -dict {PACKAGE_PIN U2 IOSTANDARD LVCMOS33} [get_ports {dip_pin[1]}]  
54 set_property -dict {PACKAGE_PIN V2 IOSTANDARD LVCMOS33} [get_ports {dip_pin[2]}]  
55 set_property -dict {PACKAGE_PIN V5 IOSTANDARD LVCMOS33} [get_ports {dip_pin[3]}]  
56 set_property -dict {PACKAGE_PIN V4 IOSTANDARD LVCMOS33} [get_ports {dip_pin[4]}]  
57 set_property -dict {PACKAGE_PIN R3 IOSTANDARD LVCMOS33} [get_ports {dip_pin[5]}]  
58 set_property -dict {PACKAGE_PIN T3 IOSTANDARD LVCMOS33} [get_ports {dip_pin[6]}]  
59 set_property -dict {PACKAGE_PIN T5 IOSTANDARD LVCMOS33} [get_ports {dip_pin[7]}]  
60  
61  
62  
63 ///////////////////////////////////////////////////////////////////LED0~LED15/////////////////////////////////////////////////////////////////  
64 set_property -dict {PACKAGE_PIN F6 IOSTANDARD LVCMOS33} [get_ports {led_pin[0]}]  
65 set_property -dict {PACKAGE_PIN G4 IOSTANDARD LVCMOS33} [get_ports {led_pin[1]}]  
66 set_property -dict {PACKAGE_PIN G3 IOSTANDARD LVCMOS33} [get_ports {led_pin[2]}]  
67 set_property -dict {PACKAGE_PIN J4 IOSTANDARD LVCMOS33} [get_ports {led_pin[3]}]  
68 set_property -dict {PACKAGE_PIN H4 IOSTANDARD LVCMOS33} [get_ports {led_pin[4]}]  
69 set_property -dict {PACKAGE_PIN J3 IOSTANDARD LVCMOS33} [get_ports {led_pin[5]}]  
70 set_property -dict {PACKAGE_PIN J2 IOSTANDARD LVCMOS33} [get_ports {led_pin[6]}]  
71 set_property -dict {PACKAGE_PIN K2 IOSTANDARD LVCMOS33} [get_ports {led_pin[7]}]  
72  
73 set_property -dict {PACKAGE_PIN K1 IOSTANDARD LVCMOS33} [get_ports {led_pin[8]}]  
74 set_property -dict {PACKAGE_PIN H6 IOSTANDARD LVCMOS33} [get_ports {led_pin[9]}]  
75 set_property -dict {PACKAGE_PIN H5 IOSTANDARD LVCMOS33} [get_ports {led_pin[10]}]  
76 set_property -dict {PACKAGE_PIN J5 IOSTANDARD LVCMOS33} [get_ports {led_pin[11]}]  
77 set_property -dict {PACKAGE_PIN K6 IOSTANDARD LVCMOS33} [get_ports {led_pin[12]}]  
78 set_property -dict {PACKAGE_PIN L1 IOSTANDARD LVCMOS33} [get_ports {led_pin[13]}]  
79 set_property -dict {PACKAGE_PIN M1 IOSTANDARD LVCMOS33} [get_ports {led_pin[14]}]  
80 set_property -dict {PACKAGE_PIN K3 IOSTANDARD LVCMOS33} [get_ports {led_pin[15]}]
```

1. 从EGo1.xdc文件中复制管脚约束信息

实验0.1：基本门电路的硬件实现



2. 将需要的管脚约束信息复制到Lab0_hdl.xdc文件中

```
C: > Xilinx > MyProject > 2024 > Lab0_hdl > Lab0_hdl.srcs > constrs_1 > new > Lab0_hdl.xdc

1
2 set_property -dict {PACKAGE_PIN T5 IOSTANDARD LVC MOS33} [get_ports {dip_pin[7]}]
3 set_property -dict {PACKAGE_PIN T3 IOSTANDARD LVC MOS33} [get_ports {dip_pin[6]}]
4
5 set_property -dict {PACKAGE_PIN K3 IOSTANDARD LVC MOS33} [get_ports {led_pin[15]}]
6
7 set_property -dict {PACKAGE_PIN R11 IOSTANDARD LVC MOS33} [get_ports {btn_pin[0]}]
8
9 set_property -dict {PACKAGE_PIN K2 IOSTANDARD LVC MOS33} [get_ports {led_pin[7]}]
10
```

4. 保存管脚约束文件

3. 将Ego1.xdc文件中的管脚名称更换成工程中的实际管脚名称

```
C: > Xilinx > MyProject > 2024 > Lab0_hdl > Lab0_hdl.srcs > constrs_1 > new > Lab0_hdl.xdc

1
2 set_property -dict {PACKAGE_PIN T5 IOSTANDARD LVC MOS33} [get_ports {and_SW0}]
3 set_property -dict {PACKAGE_PIN T3 IOSTANDARD LVC MOS33} [get_ports {and_SW1}]
4
5 set_property -dict {PACKAGE_PIN K3 IOSTANDARD LVC MOS33} [get_ports {and_LD1_0}]
6
7 set_property -dict {PACKAGE_PIN R11 IOSTANDARD LVC MOS33} [get_ports {inv_S0}]
8
9 set_property -dict {PACKAGE_PIN K2 IOSTANDARD LVC MOS33} [get_ports {inv_LD2_0}]
10
```

实验0.1：基本门电路的硬件实现



The screenshot shows the Vivado 2024.1 interface with the following details:

- Project Manager:** Sources pane shows Design Sources (1) containing Lab0_hdl (Lab0_hdl.v), Constraints (1) containing constrs_1 (Lab0_hdl.xdc), and Simulation Sources (1).
 - RTL ANALYSIS:** Sub-section under SIMULATION is highlighted with a red box.
- Source File Properties:** Lab0_hdl.v is selected. It is an Enabled Verilog file located at C:/Xilinx/MyProject/2024/Lab0_hdl/Lab0_hdl.srcs. The library is set to xil_defaultlib.
- Project Summary:** Overview tab shows the following project details:
 - Project name: Lab0_hdl
 - Project location: C:/Xilinx/MyProject/2024/Lab0_hdl
 - Product family: Artix-7
 - Project part: xc7a35tcsg324-1
 - Top module name: Lab0_hdl
 - Target language: Verilog
 - Simulator language: Mixed
- Synthesis:** Status: Not started. Messages: No errors or warnings. Part: xc7a35tcsg324-1. Strategy: Vivado Synthesis Defaults. Report Strategy: Vivado Synthesis Default Reports. Incremental synthesis: None.
- Implementation:** Status: Not started. Messages: No errors or warnings. Part: xc7a35tcsg324-1. Strategy: Vivado Implementation Defaults. Report Strategy: Vivado Implementation Default Reports. Incremental implementation: None.
- Design Runs:** Shows two runs: synth_1 and impl_1, both in the Not started state.

实验0.1：基本门电路的硬件实现



The screenshot shows the Vivado IDE interface with the following details:

- Project Manager:** The main window displays the "PROJECT MANAGER - Lab0_hdl". It shows the project structure with "Sources" containing "Design Sources (1)" (Lab0_hdl.v) and "Constraints (1)" (constrs_1).
 - Source File Properties Dialog:** A modal dialog for "Lab0_hdl.v" is open, showing the "General" tab. It includes:
 - Enabled:** Checked.
 - Location:** C:/Xilinx/MyProject
 - Type:** Verilog
 - Library:** xil_defaultlib
 - Message:** "The current Elaboration settings allow you to perform I/O planning and constraint-related work with the elaborated netlist, but these settings slow down netlist elaboration. If you are not performing I/O pin planning you can change these settings from the Elaboration page of the Project Settings dialog box."
 - Buttons:** OK (highlighted with a red box) and Cancel.
- Project Summary:** Shows the project configuration with:
 - Project name: Lab0_hdl
 - Project location: C:/Xilinx/MyProject/2024/Lab0_hdl
 - Product family: Artix-7
 - Project part: xc7a35tcsg324-1
 - Top module name: Lab0_hdl
 - Target language: Verilog
- Implementation:** Status: Not started. Messages: No errors or warnings. Part: xc7a35tcsg324-1. Strategy: Vivado Implementation Defaults. Report Strategy: Vivado Implementation Default Reports. Incremental implementation: None.
- Design Runs:** Shows two runs: synth_1 and impl_1, both in the "Not started" state.

实验0.1：基本门电路的硬件实现



國科大杭州高等研究院

Hangzhou Institute for Advanced Study, UCAS

The screenshot shows the Xilinx Vivado IDE interface with the following details:

- PROJECT MANAGER**: Settings, Add Sources, Language Templates, IP Catalog.
- IP INTEGRATOR**: Create Block Design, Open Block Design, Generate Block Design.
- SIMULATION**: Run Simulation.
- RTL ANALYSIS**: Open Elaborated Design (selected), Report Methodology, Report DRC, Report Noise, Schematic.
- SYNTHESIS**: Run Synthesis, Open Synthesized Design.
- IMPLEMENTATION**: Run Implementation, Open Implemented Design.

The central workspace displays the **Device Constraints** tab under **ELABORATED DESIGN - xc7a35tcsg324-1**. The **Schematic** tab shows two logic blocks:

- and_LD1_0_i**: An RTL AND gate with inputs `and_SW0` and `and_SW1`, outputting to `and_LD1_0_0`.
- inv_LD2_0_i**: An RTL INVERTER with input `inv_S0`, outputting to `inv_LD2_0_0`.

A red box highlights the **and_LD1_0_i** block. A red banner at the bottom right of the schematic area reads **RTL原理图与预想的设计一致** (RTL Schematic matches the expected design).

实验0.1：基本门电路的硬件实现



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

启动综合

The screenshot shows the Xilinx Vivado IDE interface. On the left, the Flow Navigator pane is open, displaying various project management options under categories like IP INTEGRATOR, SIMULATION, and RTL ANALYSIS (with Run Synthesis highlighted). The Schematic tab in the center workspace displays an RTL schematic for two logic functions:

- The first function, `and_LD1_0`, is implemented using an `RTL_AND` block. It has two inputs, `and_SW0` and `and_SW1`, and one output, `and_LD1_0`. The block has two pins, `I0` and `I1`.
- The second function, `inv_LD2_0`, is implemented using an `RTL_INV` block. It has one input, `inv_S0`, and one output, `inv_LD2_0`. The block has one pin, `I0`.

The bottom I/O Ports tab shows a table for defining I/O pins, with columns for Name, Direction, Neg Diff Pair, Package Pin, Fixed, Bank, I/O Std, Vcc0, Vref, Drive Strength, Slew Type, Pull Type, Off-Chip Termination, and IN_TERM. The table lists "All ports (5)" and "Scalar ports (5)".

实验0.1：基本门电路的硬件实现



The screenshot shows the Xilinx Vivado IDE interface. On the left, the Flow Navigator pane is open, displaying various project management options under categories like IP Catalog, IP INTEGRATOR, SIMULATION, RTL ANALYSIS, SYNTHESIS, IMPLEMENTATION, PROGRAM AND DEBUG. Under RTL ANALYSIS, 'Open Elaborated Design' is selected. In the center, the 'ELABORATED DESIGN - xc7a35tcsg324-1' window is active, showing the 'Device Constraints' tab. A 'Project Summary' card displays the project name as 'Lab0_hdl', location as 'C:/Xilinx/MyProject/2024/Lab0_hdl', product family as 'Artix-7', and part as 'xc7a35tcsg324-1'. Below this, a 'Source File Properties' dialog for 'Lab0_hdl.v' is open, showing it's a Verilog file located at 'C:/Xilinx/MyProject/2024/Lab0_hdl/Lab0_hdl' with a size of 0.7 KB. A 'Synthesis Completed' dialog box is overlaid on the interface, stating 'Synthesis successfully completed.' with 'OK' and 'Cancel' buttons. A red box highlights the 'Cancel' button. To the right of the synthesis dialog, the 'Implementation' section of the Project Summary card is visible, showing status as 'Not started', messages as 'No errors or warnings', part as 'xc7a35tcsg324-1', strategy as 'Vivado Implementation Defaults', report strategy as 'Vivado Implementation Default Reports', and incremental implementation as 'None'. At the bottom, the 'Tcl Console' and 'I/O Ports' tabs are visible.

暂不进行实现

实验0.1：基本门电路的硬件实现



File Edit Flow Tools Reports Window Layout View Help Quick Access

Synthesis Complete ✓

I/O Planning

SYNTHESIZED DESIGN - xc7a35tcsg324-1

Open Elaborated Design

Report Methodology

Report DRC

Report Noise

Schematic

SYNTHESIS

Run Synthesis

Open Synthesized Design

Constraints Wizard

Edit Timing Constraints

Set Up Debug

Report Timing Summary

Report Clock Networks

Report Clock Interaction

Report Methodology

Report DRC

Report Noise

Report Utilization

Report Power

Schematic

Internal VREF

- 0.6V
- 0.675V
- 0.75V
- 0.9V
- NONE (5)
 - I/O Bank 14

Drop I/O banks on voltages or the "NONE" folder to set/unset Internal VREF.

Source File Properties

Lab0_hdl.v

Enabled

Location: C:/Xilinx/MyProject/2024/Lab0_hdl/Lab0_hdl.srcs/s

Type: Verilog

Librav: xil_defaultlib

General Properties

Tcl Console Messages Log Reports Design Runs Package Pins I/O Ports

All ports (5)

Scalar ports (5)

Package x Device x Schematic x

7 Cells 5 I/O Ports 10 Nets

and_SW0 and_SW1 inv_S0

and_SW0_IBUF_inst and_SW1_IBUF_inst inv_S0_IBUF_inst

and_LD1_0_OBUF_inst_i_1 inv_LD2_0_OBUF_inst_i_1

LUT2 LUT1

and_LD1_0_OBUF_inst inv_LD2_0_OBUF_inst

and_LD1_0 and_LD1_0_OBUF and_LD1_0_OBUF_inv LD2_0

2. 综合后，门电路变成了LUT，插入了输入、输出buf
(可以通过LUT的真值表确认电路功能正确性)

实验0.1：基本门电路的硬件实现



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

File Edit Flow Tools Reports Window Layout View Help Quick Access

SYNTHESIZED DESIGN - xc7a35tcsq324-1

Sources Netlist Device Constraints x ? □ Schematic x ? □

Internal VREF
0.6V
0.675V
0.75V
0.9V

NONE (5)
I/O Bank 14

Report DRC
Report Noise
Schematic
SYNTHESIS
Run Synthesis
Open Synthesized Design
Constraints Wizard
Edit Timing Constraints
Set Up Debug
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Report DRC
Report Noise
Report Utilization
Report Power
Schematic
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager

Package x Device x Schematic x ? □

7 Cells 5 I/O Ports 10 Nets

Launch Runs
Launch the selected synthesis or implementation runs.
Launch directory: <Default Launch Directory>
Options
Launch runs on local host: Number of jobs: 6
Generate scripts only
Don't show this dialog again
OK Cancel

Tcl Console Messages Log Reports Design Runs Package Pins I/O Ports

Name Direction Neg Diff Pair Package Pin Fixed Bank I/O Std Vcco Vref Drive Strength Slew Type Pull Type Off-Chip Termination IN_TERM

Scalar ports (5)

1.启动实现

实验0.1：基本门电路的硬件实现



國科大杭州高等研究院

Hangzhou Institute for Advanced Study, UCAS

The screenshot shows the Xilinx Vivado IDE interface with the following details:

- File Navigator**: Shows report methodology, DRC, noise, and schematic.
- SYNTHESIS**: Run Synthesis, Open Synthesized Design (Constraints Wizard, Edit Timing Constraints, Set Up Debug, Report Timing Summary, Report Clock Networks, Report Clock Interaction, Report Methodology, Report DRC, Report Noise, Report Utilization, Report Power).
- IMPLEMENTATION**: Run Implementation, Open Implemented Design.
- PROGRAM AND DEBUG**: Generate Bitstream, Open Hardware Manager.

The main workspace displays the **SYNTHESIZED DESIGN - xc7a35tcsg324-1** project. The **Device Constraints** tab is active, showing internal VREF values (0.6V, 0.675V, 0.75V, 0.9V). The **Schematic** tab shows a logic diagram for the `and_LD1_0_OBUF_inst_i_1` and `inv_LD2_0_OBUF_inst_i_1` instances. The `and_LD1_0_OBUF_inst_i_1` instance uses LUT2 and OBUF components, while the `inv_LD2_0_OBUF_inst_i_1` instance uses LUT1 and OBUF components. A red box highlights the "Open Implemented Design" button in the implementation completion dialog.

Implementation Completed Dialog:

- Message: Implementation successfully completed.
- Buttons:
 - OK** (highlighted with a red box)
 - Cancel**
- Options:
 - Open Implemented Design** (radio button selected, highlighted with a red box)
 - Generate Bitstream**
 - View Reports**
 - Don't show this dialog again

查看实现后的电路 (View Implemented Design) is written in large red text at the bottom right of the schematic area.

实验0.1：基本门电路的硬件实现



SYNTHESIZED DESIGN - xc7a35tcsg324-1

Implementation Complete ✓

I/O Planning

SYNTHESIS

- Run Synthesis
- Open Synthesized Design
 - Constraints Wizard
 - Edit Timing Constraints
 - Set Up Debug
 - Report Timing Summary
 - Report Clock Networks
 - Report Clock Interaction
 - Report Methodology
 - Report DRC
 - Report Noise
 - Report Utilization
 - Report Power
- Schematic

Open Implemented Design

Generate Bitstream

Open Hardware Manager

Device Constraints

Internal VREF

- 0.6V
- 0.675V
- 0.75V
- 0.9V

Drop I/O banks on voltages or the "NONE" folder to set/unset Internal VREF.

Source File Properties

Lab0_hdl.v

Enabled

Location: C:/Xilinx/MyProject/202

Type: Verilog

Library: xil_defaultlib

Size: 0.7 KB

Do you want to close 'Synthesized Design' before opening 'Implemented Design'?

Yes

No

Cancel

Tcl Console

Messages

Log

Reports

Design Runs

Package Pins

I/O Ports

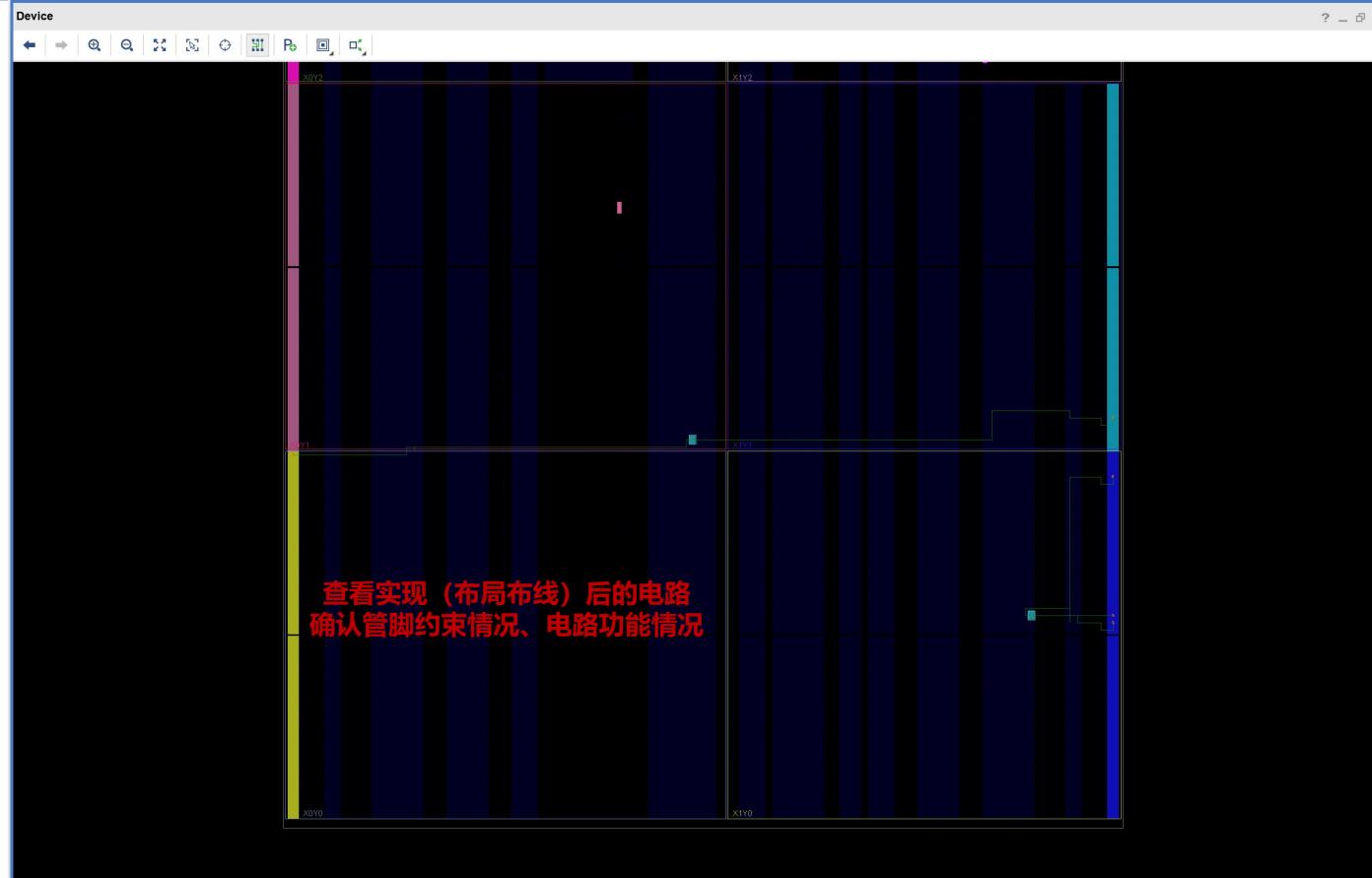
Name Direction Neg Diff Pair Package Pin Fixed Bank I/O Std Vcco Vref Drive Strength Slew Type Pull Type Off-Chip Termination IN_TERM

All ports (5)

Scalar ports (5)

The screenshot shows the Xilinx Vivado IDE interface. The top menu bar includes File, Edit, Flow, Tools, Reports, Window, Layout, View, Help, and Quick Access. A central workspace displays a schematic diagram of a logic circuit. The circuit consists of several logic blocks: an IBUF (Input Buffer) labeled 'and_SW0_IBUF_inst', two LUT2 blocks labeled 'and_LD1_0_LUT2_inst_i_1' and 'and_LD1_0_LUT2_inst_i_2', and two OBIF (Output Buffer) blocks labeled 'and_LD1_0_OBUF_inst' and 'inv_LD2_0_OBUF_inst'. The 'and_LD1_0_OBUF_inst' block has its output connected to the 'and_LD1_0' port. The 'and_LD1_0_LUT2_inst_i_1' block has its output connected to the 'I0' port of the 'and_LD1_0_OBUF_inst' block. The 'and_LD1_0_LUT2_inst_i_2' block has its output connected to the 'I1' port of the 'and_LD1_0_OBUF_inst' block. The 'and_LD1_0_OBUF_inst' block also has its output connected to the 'and_LD1_0' port. The 'inv_LD2_0_OBUF_inst' block has its output connected to the 'inv_LD2_0' port. The 'and_LD1_0_LUT2_inst_i_1' block has its output connected to the 'I0' port of the 'inv_LD2_0_OBUF_inst' block. The 'inv_LD2_0_OBUF_inst' block has its output connected to the 'inv_LD2_0' port. A 'Source File Properties' dialog box is open over the schematic, asking if the user wants to close the 'Synthesized Design' before opening 'Implemented Design'. The 'Yes' button is highlighted with a red border. The dialog also shows the source file 'Lab0_hdl.v' is enabled, located at 'C:/Xilinx/MyProject/202', and is a Verilog file in the 'xil_defaultlib' library. The file size is 0.7 KB. The 'General' tab is selected in the properties dialog. At the bottom of the interface, there is a table titled 'I/O Ports' with columns for Name, Direction, Neg Diff Pair, Package Pin, Fixed, Bank, I/O Std, Vcco, Vref, Drive Strength, Slew Type, Pull Type, Off-Chip Termination, and IN_TERM. The table lists 'All ports (5)' and 'Scalar ports (5)'.

实验0.1：基本门电路的硬件实现



实验0.1：基本门电路的硬件实现



1. 查看工程信息

Implementation Complete ✓
Default Layout

IMPLEMENTED DESIGN - xc7a35tcsg324-1

Project Summary | Device | Overview | Dashboard

Settings Edit

Project name: Lab0_hdl
Project location: C:/Xilinx/MyProject/2024/Lab0_hdl
Product family: Artix-7
Project part: xc7a35tcsg324-1
Top module name: Lab0_hdl
Target language: Verilog
Simulator language: Mixed

Source File Properties

Lab0_hdl.v
Enabled
Location: C:/Xilinx/MyProject/2024/Lab0_hdl/Lab0_h
Type: Verilog
Library: xil_defaultlib
Size: 0.7 KB

Synthesis

Status: Complete
Messages: 1 warning
Part: xc7a35tcsg324-1
Strategy: Vivado Synthesis Defaults
Report Strategy: Vivado Synthesis Default Reports
Incremental synthesis: None

Implementation

Status: Complete
Messages: 3 warnings
Part: xc7a35tcsg324-1
Strategy: Vivado Implementation Defaults
Report Strategy: Vivado Implementation Default Reports
Incremental implementation: None

DRC Violations

Tcl Console Messages Log Reports Design Runs Power DRC Timing

Warning (5) Info (110) Status (225)

2. 重视各流程中出现的警告信息

3. 电路中没有时钟，会出现相应警告信息

[Place 46-29] place_design is not in timing mode. Skip physical synthesis in placer

[Power 33-232] No user defined clocks were found in the design! Power estimation will be inaccurate until this is corrected.
Resolution: Please specify clocks using create_clock/create_generated_clock for sequential elements. For pure combinatorial circuits, please specify a virtual clock, otherwise the vectorless estimation might be inaccurate

实验0.1：基本门电路的硬件实现



The screenshot shows the Xilinx Vivado IDE interface with the following details:

- File Navigator**: Shows various report options like Report Clock Networks, Report DRC, etc.
- IMPLEMENTATION**:
 - Run Implementation
 - Open Implemented Design**:
 - Constraints Wizard
 - Edit Timing Constraints
 - Report Timing Summary
 - Report Clock Networks
 - Report Clock Interaction
 - Report Methodology
 - Report DRC
 - Report Noise
 - Report Utilization
 - Report Power
 - Schematic

- PROGRAM AND DEBUG**:
- Generate Bitstream** (highlighted with a red box)
- Open Hardware Manager
- IMPLEMENTED DESIGN - xc7a35tcsg324-1**:
- Sources**: Netlist (Lab0_hdl)
- Project Summary**: Project name: Lab0_hdl, Project location: C:/Xilinx/MyProject/2024/Lab0_hdl, Product family: Artix-7
- Implementation**: Status: Complete, Messages: 3 warnings, Part: xc7a35tcsg324-1, Strategy: Vivado Implementation Defaults, Report Strategy: Vivado Implementation Default Reports, Incremental implementation: None
- Timing**: Setup | Hold | Pulse Width
- Launch Runs**: A dialog box is open, prompting to "Launch the selected synthesis or implementation runs". It includes:
- Launch directory: <Default Launch Directory>
- Options:
 - Enabled
 - Location: C:/Xilinx/MyProject/2024/Lab0_hdl
 - Type: Verilog
 - Library: xil_defaultlib
 - Size: 0.7 KB
- Buttons: OK (highlighted with a red box) and Cancel
- Tcl Console**: Shows the following messages:
- Messages**: Warning (5), Info (110), Status (225)
- Synthesis**: 1 warning
 - [Constraints 18-5210] No constraints selected for write.
Resolution: This message can indicate that there are no constraints for the design, or it can indicate that the used_in_flags are set such that the constraints are ignored. This latter case is used when running synth_design to not write synthesis constraints to the resulting checkpoint. Instead, project constraints are read when the synthesized design is opened.
- Implementation**: 3 warnings
 - Place Design (1 warning)
 - [Constraints 18-29] place_design is not in timing mode. Skip physical synthesis in placer
 - Route Design (2 warnings)

1.生成bit文件

实验0.1：基本门电路的硬件实现



File Edit Flow Tools Reports Window Layout View Help Quick Access

IMPLEMENTED DESIGN - xc7a35tcsg324-1

Report Clock Networks
Report Clock Interaction
 Report Methodology
Report DRC
Report Noise
Report Utilization
 Report Power
 Schematic

IMPLEMENTATION

Run Implementation
 Open Implemented Design
Constraints Wizard
Edit Timing Constraints
 Report Timing Summary
Report Clock Networks
Report Clock Interaction
 Report Methodology
Report DRC
Report Noise
Report Utilization
 Report Power
 Schematic

Generate Bitstream
 Open Hardware Manager

Sources Netlist Project Summary Device Overview | Dashboard

Lab0_hdl
Nets (10)
Leaf Cells (7)

Settings Edit

Project name: Lab0_hdl
Project location: C:/Xilinx/MyProject/2024/Lab0_hdl
Product family: Artix-7
Project part: xc7a35tcsg324-1

Bitstream Generation Completed

Source File Properties

Lab0_hdl.v
Enabled
Location: C:/Xilinx/MyProject/2024/Lab0_hdl/Lab0_hdl.v
Type: Verilog
Library: xil_defaultlib
Size: 0.7 KB

General Properties

Bitstream Generation successfully completed.

Next

View Reports
 Open Hardware Manager
 Generate Memory Configuration File
 Don't show this dialog again

OK Cancel

Implementation Summary | Route Status

Status: Complete
Messages: 4 warnings
Part: xc7a35tcsg324-1
Strategy: Vivado Implementation Defaults
Report Strategy: Vivado Implementation Default Reports
Incremental implementation: None

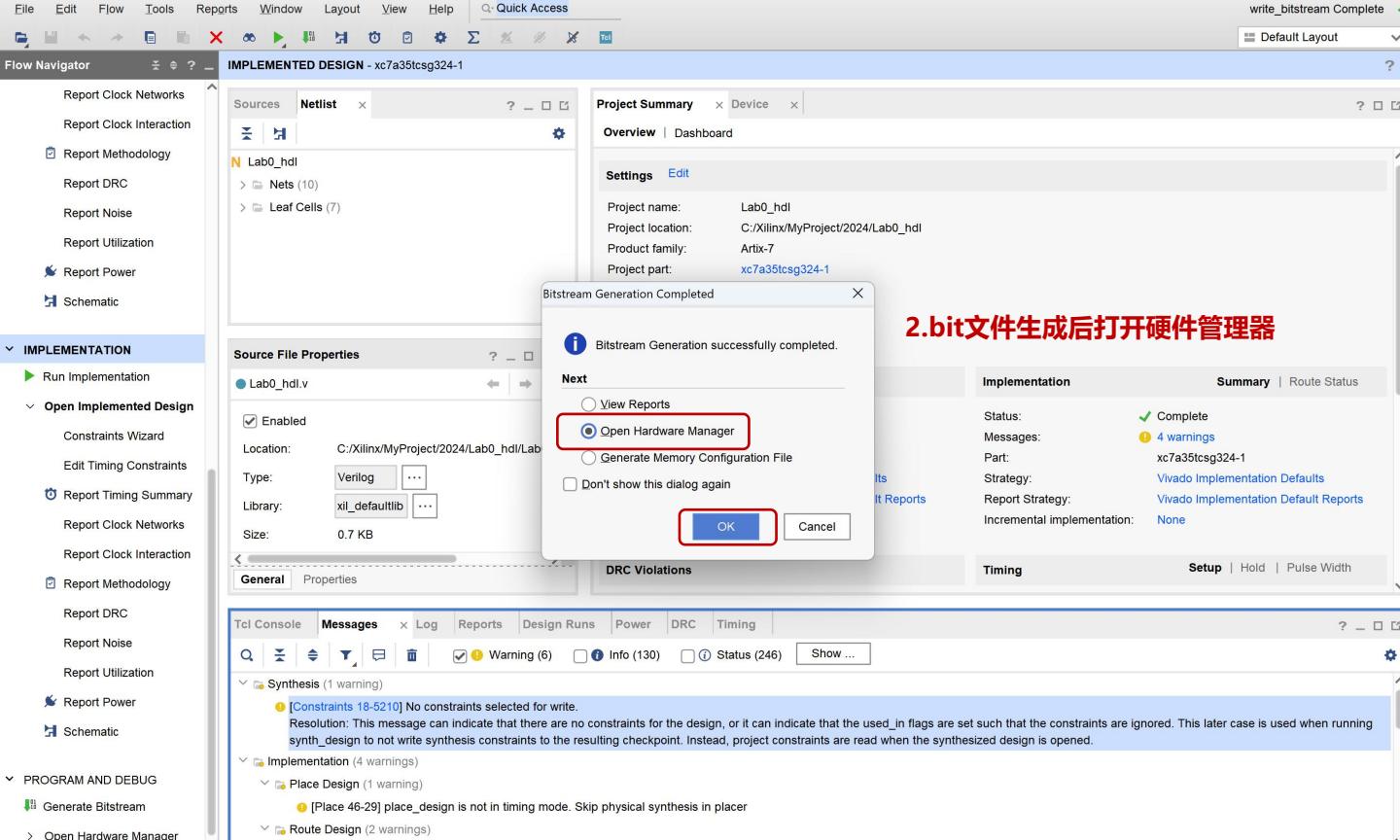
Tcl Console Messages Log Reports Design Runs Power DRC Timing

Warning (6) Info (130) Status (246) Show ...

Synthesis (1 warning)
[Constraints 18-5210] No constraints selected for write.
Resolution: This message can indicate that there are no constraints for the design, or it can indicate that the used_in flags are set such that the constraints are ignored. This later case is used when running synth_design to not write synthesis constraints to the resulting checkpoint. Instead, project constraints are read when the synthesized design is opened.

Implementation (4 warnings)
Place Design (1 warning)
[Place 46-29] place_design is not in timing mode. Skip physical synthesis in placer
Route Design (2 warnings)

2.bit文件生成后打开硬件管理器



实验0.1：基本门电路的硬件实现



The screenshot shows the Vivado 2019.1 software interface. On the left, the Flow Navigator pane is open, displaying various project management options like Report Clock Interaction, Report Methodology, and Implementation. The Implementation section is expanded, showing sub-options such as Run Implementation, Open Implemented Design, and Program Device. The Program Device option is highlighted with a red box. In the center, the HARDWARE MANAGER window is active, showing a list of hardware devices. One device, 'localhost (1) Connected' under 'xilinx_tcf/XI Open', is selected and has a context menu open. The 'Program Device...' option in this menu is also highlighted with a red box. To the right of the manager, a message bar indicates 'write_bitstream Complete' with a checkmark. At the bottom, the Tcl Console window displays command-line output related to hardware target configuration.

3.对FPGA进行下载

```
open_hw_target
INFO: [Labtoolscl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Xilinx/1234-tula
set_property PROGRAM.FILE {C:/Xilinx/MyProject/2024/Lab0_hdl/Lab0_hdl.runs/impl_1/Lab0_hdl.bit} [get_hw_devices xc7a35t_0]
current_hw_device [get_hw_devices xc7a35t_0]
refresh_hw_device -update_hw_probes false [lindex [get_hw_devices xc7a35t_0] 0]
INFO: [Labtools 27-1434] Device xc7a35t (JTAG device index = 0) is programmed with a design that has no supported debug core(s) in it.
```

实验0.1：基本门电路的硬件实现



The screenshot shows the Xilinx Vivado interface. On the left, the Flow Navigator pane is open, displaying various project management options under categories like Report Clock Interaction, Implementation, and Program and Debug. In the center, the Hardware Manager window is active, showing a list of hardware devices connected to 'localhost'. A 'Program Device' dialog box is overlaid on the main window, prompting the user to select a bitstream file ('_1/Lab0_hd1.bit') and click the 'Program' button. The bottom of the interface features a Tcl Console window showing the command-line interaction for opening the hardware target and setting the bitstream file. A red box highlights the 'Program' button in the dialog box.

4.确认bit文件后对器件进行下载

实验0.1：基本门电路的硬件实现



■实验效果

• 按键S0按下，LD2_0熄灭，按键S0松开，LD2_0点亮

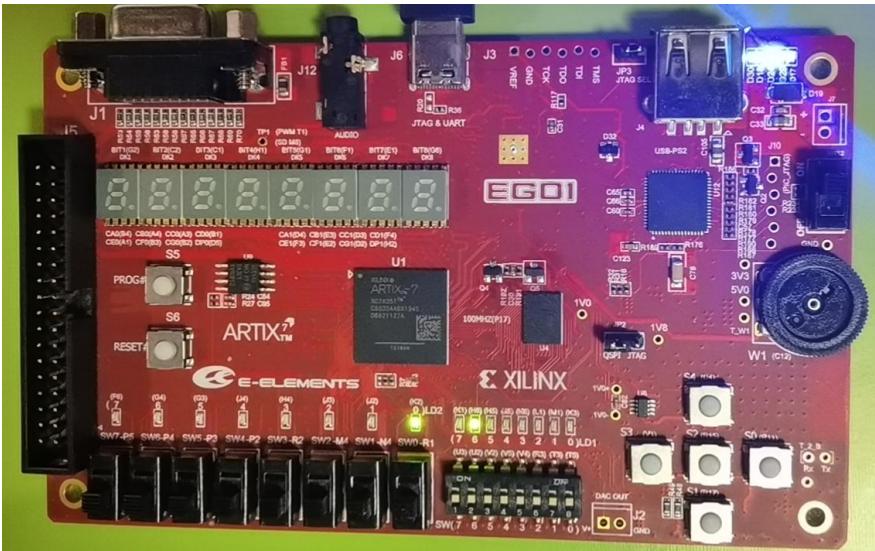
• DIP8：

SW0和SW1均为ON，LD1_0点亮

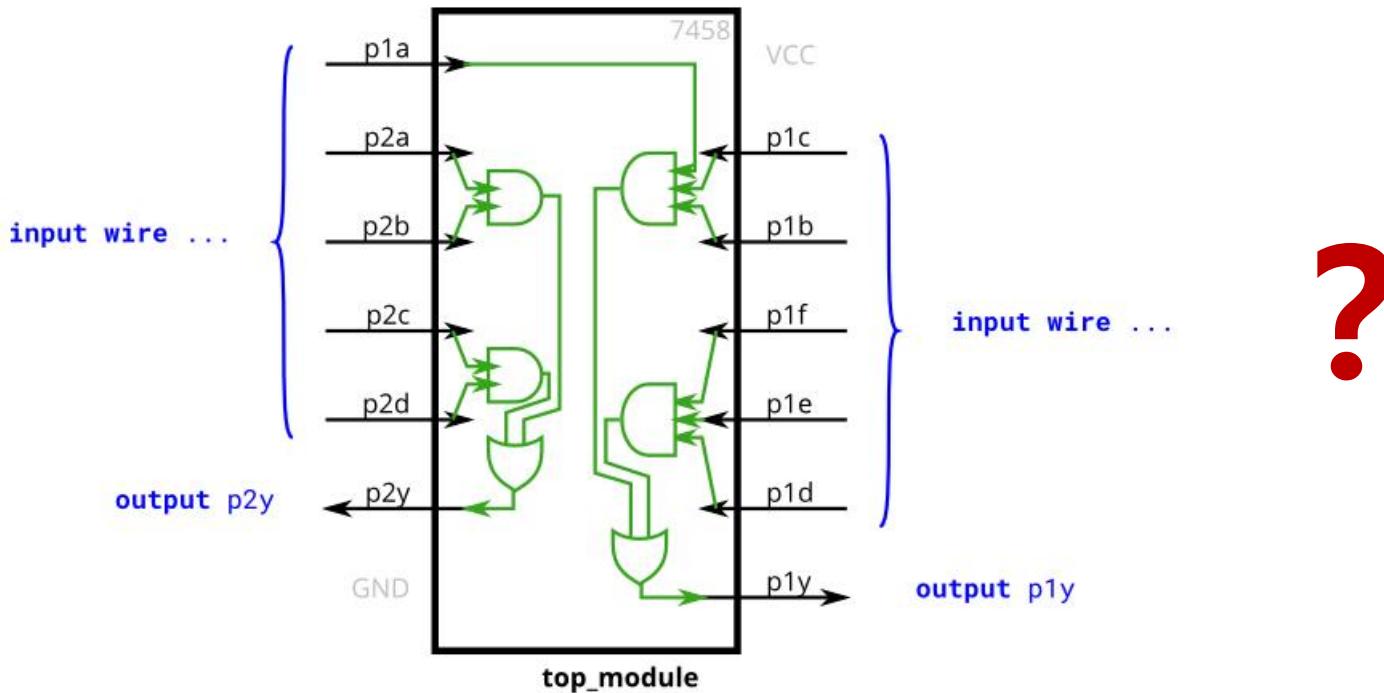
SW2或SW3一个为ON，LD1_2点亮

SW4和SW5不同，LD1_4点亮

SW6和SW7相同，LD1_6点亮



实验0.1：基本门电路的硬件实现



<https://hdlbits.01xz.net/wiki/7458>

课后作业：HDLbits题目



▶ Getting Started

▼ Verilog Language

▶ Basics

▼ Vectors

Vectors

- Vectors in more detail
- Vector part select

Bitwise operators

Four-input gates

Vector concatenation operator

Vector reversal 1

Replication operator

More replication

▶ Modules: Hierarchy

▶ Procedures

▶ More Verilog Features

▶ Circuits

▶ Verification: Reading Simulations

▶ Verification: Writing Testbenches

▶ CS450

▶ Getting Started

▼ Verilog Language

▶ Basics

▶ Vectors

▼ Modules: Hierarchy

Modules

- Connecting ports by position
- Connecting ports by name
- Three modules

Modules and vectors

Adder 1

Adder 2

Carry-select adder

Adder-subtractor

▶ Procedures

▶ More Verilog Features

▶ Circuits

▶ Verification: Reading Simulations

▶ Verification: Writing Testbenches

▶ CS450

▶ Getting Started

▼ Verilog Language

▶ Basics

▶ Vectors

▶ Modules: Hierarchy

▼ Procedures

Always blocks (combinational)

Always blocks (clocked)

If statement

If statement latches

Case statement

Priority encoder

Priority encoder with casez

Avoiding latches

▶ More Verilog Features

▶ Circuits

▶ Verification: Reading Simulations

▶ Verification: Writing Testbenches

▶ CS450

下节预告



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Lab1实验内容

Lab1LED控制实验



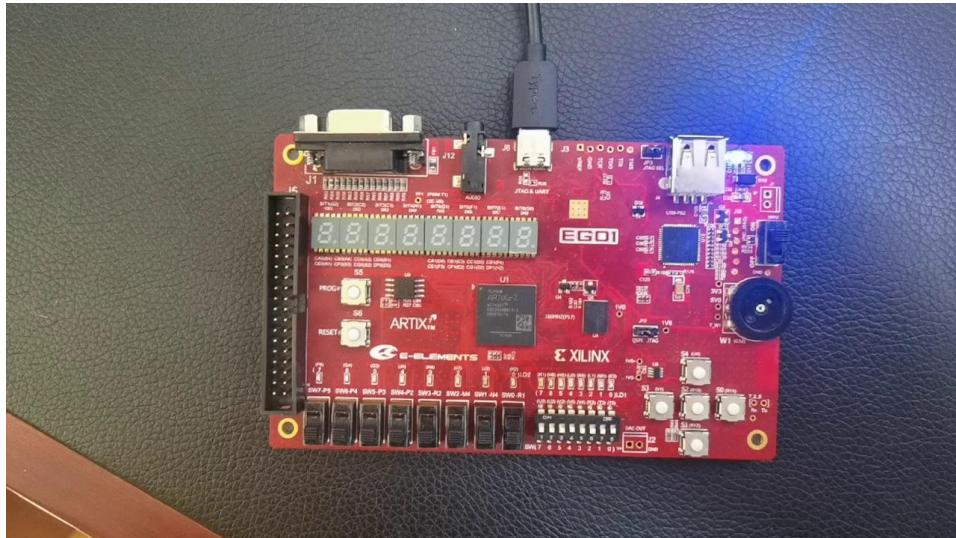
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■实验内容

- 设计并实现一套简易双控灯
- 三个按键：总控开关、灯控开关A、灯控开关B
- 一个LED
- 总控开关实现LED的整体控制
- 灯控开关A&B实现LED的双控

■实验目的

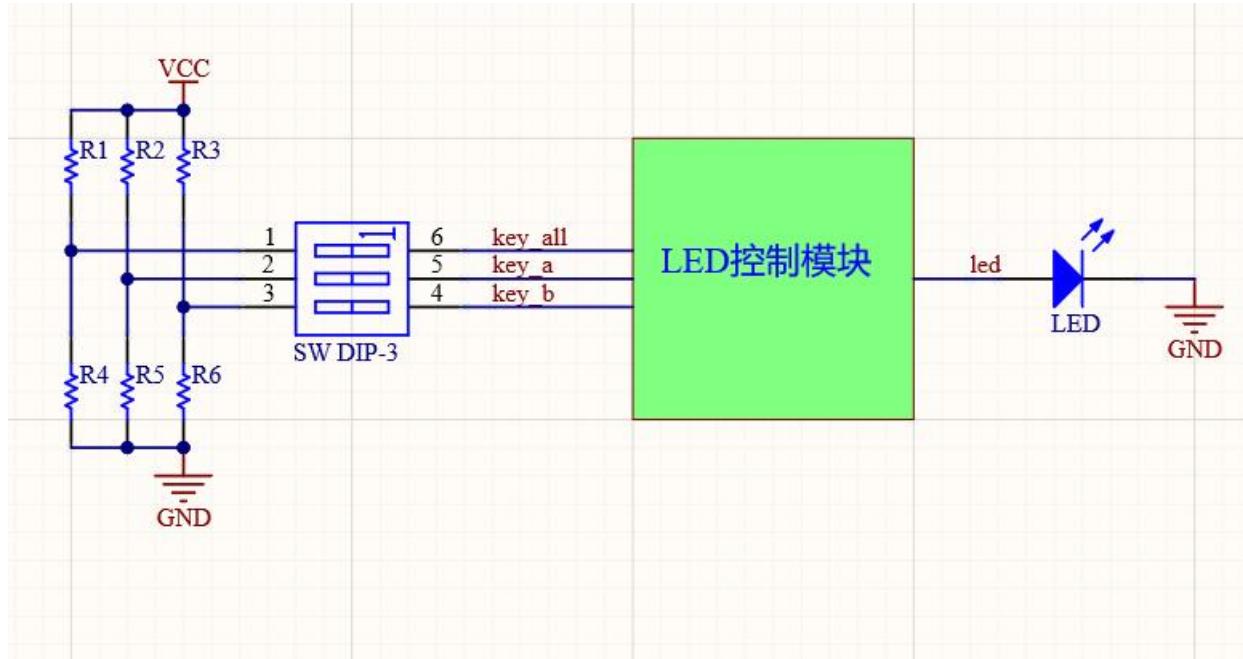
- 进一步熟悉Vivado设计流程
- 熟悉Verilog程序框架
- 学会Verilog的 always,if else语句
- 学会使用always@(*)语法实现组合逻辑电路
- 了解Testbench的使用



Lab1 LED控制实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



LED控制实验-电路图

Lab1LED控制实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

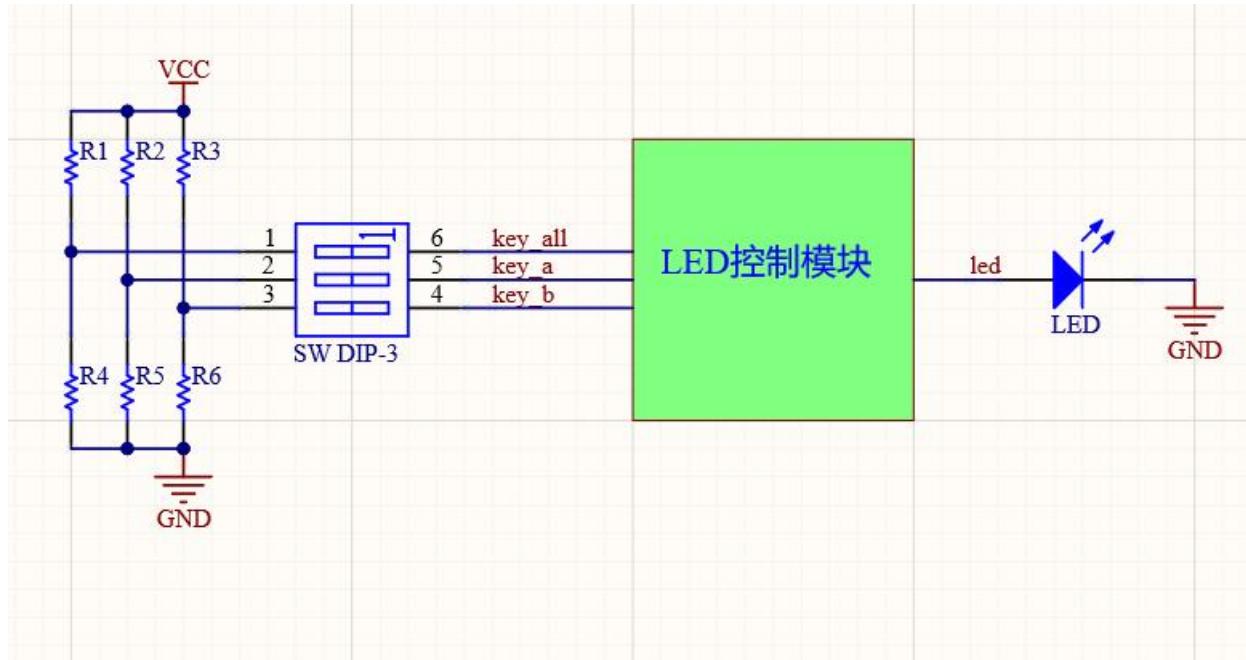
key_all	key_a	key_b	led
0	X	X	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

led控制实验-真值表

Lab1 LED控制实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



LED控制实验-电路图

Lab1LED控制实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

key_all	key_a	key_b	led
0	X	X	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

led控制实验-真值表



THANKS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第三讲 (3.1 FPGA硬件电路)

2025年3月18日

Xilinx 7系列FPGA



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

7 Series Product Selection Guide

■ 7系列FPGA

- Virtex7
- Kintex7
- Artix7
- Spartan7



45nm

SPARTAN⁷

28nm

VIRTEX⁷
KINTEX⁷
ARTIX⁷
SPARTAN⁷

20nm

VIRTEX⁷
UltraSCALE
KINTEX⁷
UltraSCALE

16nm

VIRTEX⁷
UltraSCALE+
KINTEX⁷
UltraSCALE+
ARTIX⁷
UltraSCALE+

SPARTAN⁷ ARTIX⁷ KINTEX⁷ VIRTEX⁷

XILINX



7系列FPGA 文档资料

References

[DS180](#), 7 Series FPGAs Overview

[DS181](#), Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics

[DS182](#), Kintex-7 FPGAs Data Sheet: DC and AC Switching Characteristics

[DS183](#), Virtex-7 T and XT FPGAs Data Sheet: DC and AC Switching Characteristics

[UG470](#), 7 Series FPGAs Configuration User Guide

[UG471](#), 7 Series FPGAs SelectIO Resources User Guide

[UG472](#), 7 Series FPGAs Clocking Resources User Guide

[UG473](#), 7 Series FPGAs Memory Resources User Guide

[UG474](#), 7 Series FPGAs Configurable Logic Block User Guide

[UG475](#), 7 Series FPGAs Packaging and Pinout User Guide

[UG476](#), 7 Series FPGAs GTX/GTH Transceivers User Guide

[UG479](#), 7 Series FPGAs DSP48E1 Slice User Guide

[UG480](#), 7 Series FPGAs and Zynq-7000 All Programmable SoC XADC Dual 12-Bit 1 MSPS ADC User Guide

[UG482](#), 7 Series FPGAs GTP Transceivers User Guide

[UG483](#), 7 Series FPGAs PCB Design Guide

Xilinx 7系列FPGA



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Vivado 2019.1

File Flow Tools Window Help Q: Quick Access

Documentation and Tutorials...

Quick Take Videos...
Release Notes Guide...
Design Hubs
Quick Help... F1
Leave Feedback...
Search Answer Records...
Get Technical Support...
Add Design Tools or Devices...
Manage License...
Obtain a License Key...
Vivado on the Web...
About Vivado...

文档和向导

VIVADO
HLx Editions

Quick Start

Create Project >
Open Project >
Open Example Project

Tasks

Manage IP >
Open Hardware Manager >
Xilinx Tcl Store >

Learning Center

Documentation and Tutorials >
Quick Take Videos >
Release Notes Guide >

Recent Projects

Lab01_text
C:/Xilinx/MyProject_2022/Lab01_text
Lab00_sch
C:/Xilinx/MyProject_2022/Lab00_sch
Lab3_seg7_dynamic
C:/Xilinx/MyProject/Lab3_seg7_dynamic

XILINX

Xilinx 7系列FPGA



國科大杭州高華研究院 Hangzhou Institute for Advanced Study, UCAS

文档筛选

文档编号

TOC	Doc ID	Version	MB	Published	Status
	DS180				Current
	DS185				Current
	DS197				Current
	XMP101				Current
	XMP086				Current
	XMP085				Current
	XMP084				Current
	DS181				Current
	DS182				Current
	DS189				Current
	DS183				Current
	DS593				Current
	UG473				Current
	UG474				1.8
	UG471				Current
	UG472				4.3
	UG470				5.7
	UG479				Current
	UG476				Current
	UG482				Current
	UG480				Current
	UG483				3.5
	UG475				Current
	UG1099				Current
	UG429				Current
	UG116				Current

Xilinx 7系列FPGA



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Table 1: 7 Series Families Comparison

1个Logic Cell= 1个 4输入LUT+1个触发器

Max. Capability	Spartan-7	Artix-7	Kintex-7	Virtex-7
Logic Cells	102K	215K	478K	1,955K
Block RAM ⁽¹⁾	4.2 Mb	13 Mb	34 Mb	68 Mb
DSP Slices	160	740	1,920	3,600
DSP Performance ⁽²⁾	176 GMAC/s	929 GMAC/s	2,845 GMAC/s	5,335 GMAC/s
MicroBlaze CPU ⁽³⁾	260 DMIPs	303 DMIPs	438 DMIPs	441 DMIPs
Transceivers	–	16	32	96
Transceiver Speed	–	6.6 Gb/s	12.5 Gb/s	28.05 Gb/s
Serial Bandwidth	–	211 Gb/s	800 Gb/s	2,784 Gb/s
PCIe Interface	–	x4 Gen2	x8 Gen2	x8 Gen3
Memory Interface	800 Mb/s	1,066 Mb/s	1,866 Mb/s	1,866 Mb/s
I/O Pins	400	500	500	1,200
I/O Voltage	1.2V–3.3V	1.2V–3.3V	1.2V–3.3V	1.2V–3.3V
Package Options	Low-Cost, Wire-Bond	Low-Cost, Wire-Bond, Bare-Die Flip-Chip	Bare-Die Flip-Chip and High- Performance Flip-Chip	Highest Performance Flip-Chip

7系列FPGA资源比较

Xilinx 7系列FPGA



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Artix-7 FPGA Feature Summary

Table 4: Artix-7 FPGA Feature Summary by Device

Device	Logic Cells	Configurable Logic Blocks (CLBs)		DSP48E1 Slices ⁽²⁾	Block RAM Blocks ⁽³⁾			CMTs ⁽⁴⁾	PCIe ⁽⁵⁾	GTPs	XADC Blocks	Total I/O Banks ⁽⁶⁾	Max User I/O ⁽⁷⁾
		Slices ⁽¹⁾	Max Distributed RAM (Kb)		18 Kb	36 Kb	Max (Kb)						
XC7A12T	12,800	2,000	171	40	40	20	720	3	1	2	1	3	150
XC7A15T	16,640	2,600	200	45	50	25	900	5	1	4	1	5	250
XC7A25T	23,360	3,650	313	80	90	45	1,620	3	1	4	1	3	150
XC7A35T	33,280	5,200	400	90	100	50	1,800	5	1	4	1	5	250
XC7A50T	52,160	8,150	600	120	150	75	2,700	5	1	4	1	5	250
XC7A75T	75,520	11,800	892	180	210	105	3,780	6	1	8	1	6	300
XC7A100T	101,440	15,850	1,188	240	270	135	4,860	6	1	8	1	6	300
XC7A200T	215,360	33,650	2,888	740	730	365	13,140	10	1	16	1	10	500

Artix-7系列 FPGA资源比较

口袋实验室-硬件组成

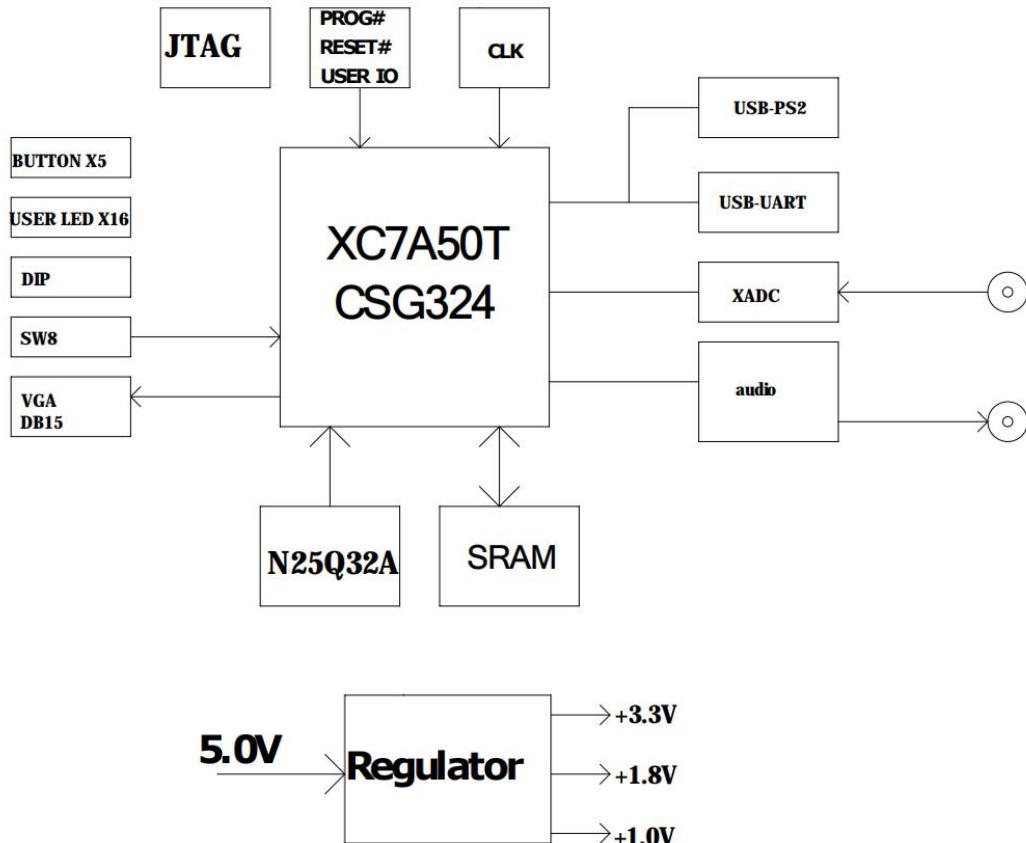


■ FPGA硬件能正常工作的必要条件

- FPGA芯片
- 电源
- 配置电路
- 时钟
- 复位电路

■ FPGA硬件其他外围电路

- 与硬件设计相关
- SDRAM、FLASH等存储器
- 接口 (rs232、网口、spi、iic等)
- IO控制
- 数据采集 (ADC、DAC)
- 人机交互 (显示屏、键盘鼠标)

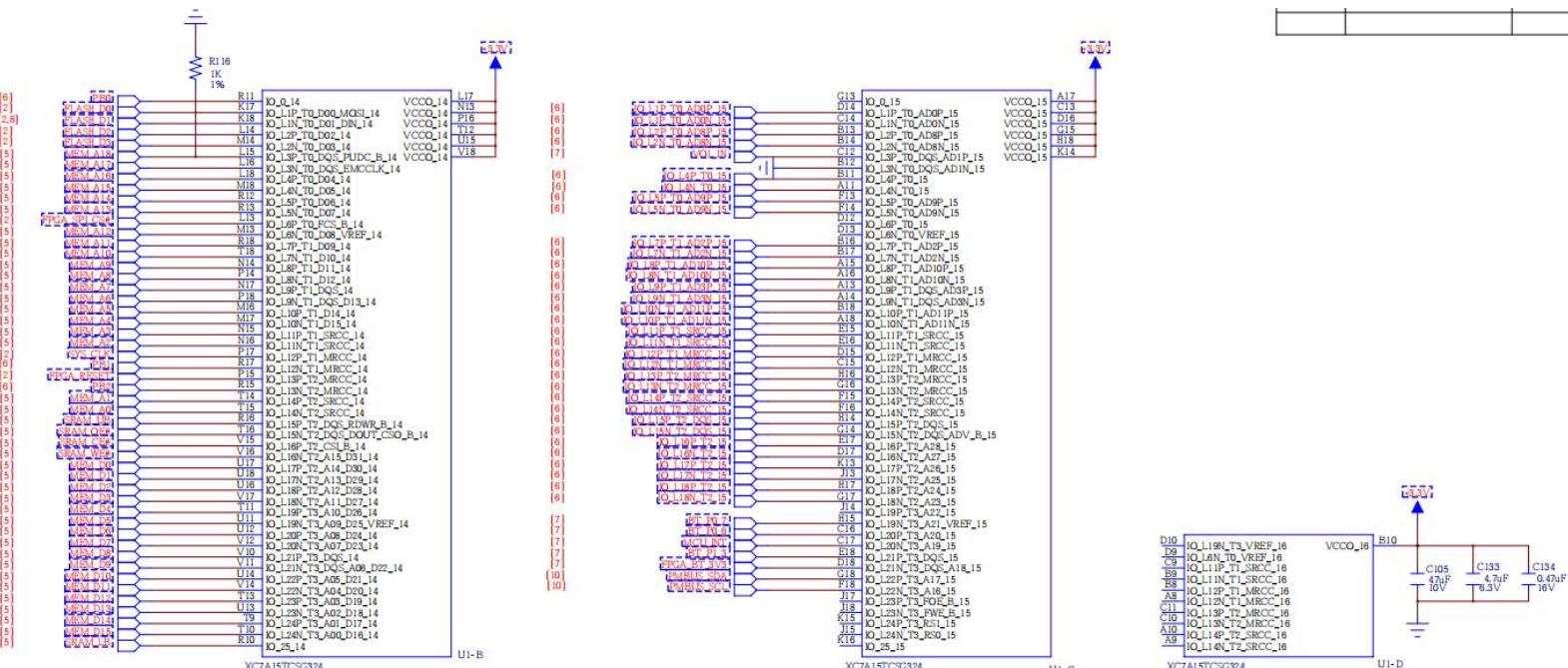


口袋实验室介绍



國科大杭州高等研究院

Hangzhou Institute for Advanced Study, UCAS



FPGA I/O电路

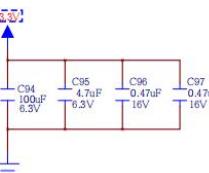
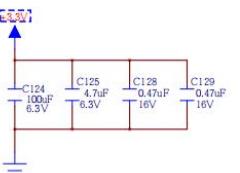
口袋实验室介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA I/O电路



口袋实验室介绍

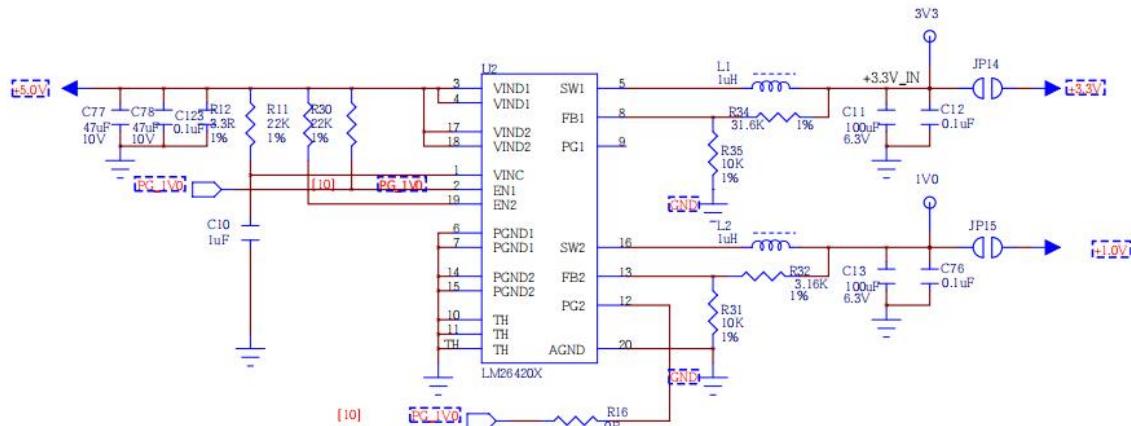


國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ Artix-7 供电要求

- VCCINT 1.0V
- VCCAUX 1.8V
- VCCBRAM 1.0V
- VCCO 1.14~3.465V
- VCCADC 1.8V

- 上电顺序: VCCINT->VCCBRAM->VCCAUX->VCCO
- 掉电顺序: 与上电顺序相反
- VCCINT与VCCBRAM电压相同, 可共用同一电源并同时上电
- 若VCCAUX与VCCO电压相同(如均为1.8V), 也可合并供电
- EGo1上电成功后D18点亮



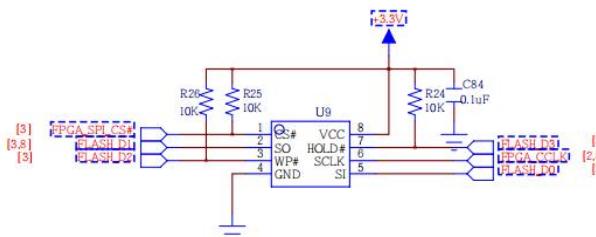
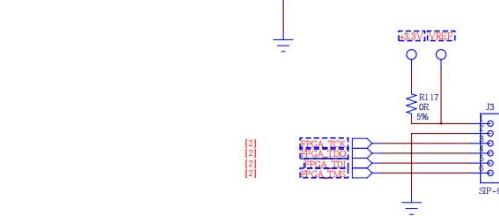
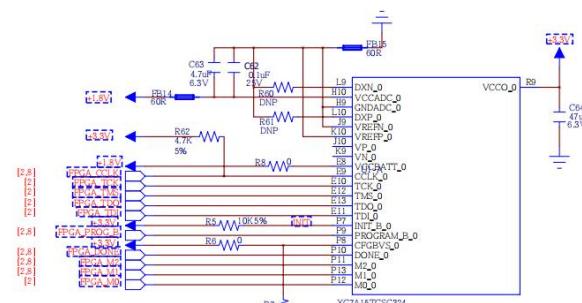
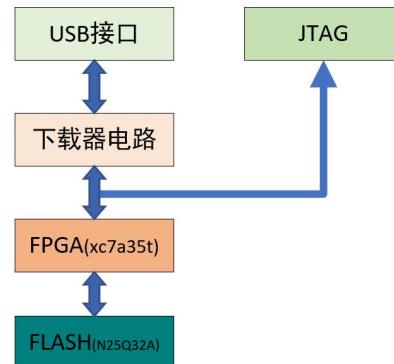
口袋实验室介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ Artix-7配置电路

- FPGA转入工作模式必须要进行配置
- EGO1主要通过J6(TypeC接口)进行配置
- 也可以通过J3(JTAG接口)进行配置
- 板载SPI Flash可掉电保存配置程序，实现上电自动加载
- 配置成功后D24点亮

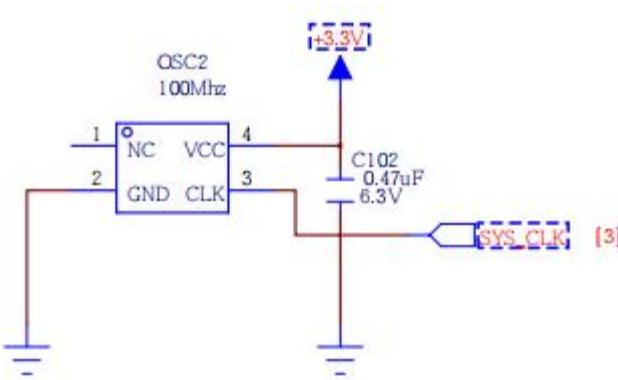


FPGA 配置电路



■ Artix-7时钟电路

- 采用100MHz有源晶振
- 接入全局时钟网络(FPGA P17引脚)



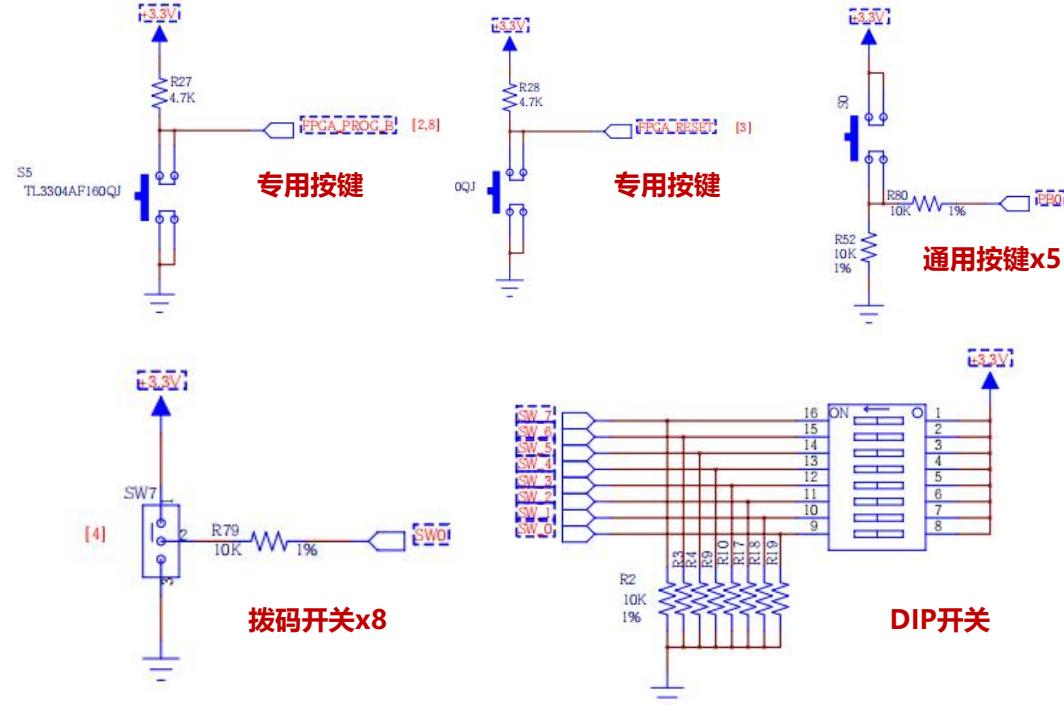
FPGA 时钟电路

口袋实验室介绍



■按键

- 按键S5擦除FPGA配置
- 按键S6提供系统外部复位信号，低电平有效
S6->P15
- S0~S4五个通用按键，点触式，高电平有效
S0->R11,S1->R17,S2->R15,S3->V1,S4->U4
- 8个拨码开关，ON->高电平，OFF->低电平
SW0->R1,SW1->N4,SW2->M4,SW3->R2
SW4->P2,SW5->P3,SW6->P4,SW7->P5
- 8位DIP开关，ON->高电平，OFF->低电平
SW0->T5,SW1->T3,SW2->R4,SW3->V4
SW4->V5,SW5->V2,SW6->U2,SW7->U3



EOG1按键资源

口袋实验室介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ LED

- 16个LED

LD1_0->K3, LD1_1->M1, LD1_2->L1, LD1_3->K6

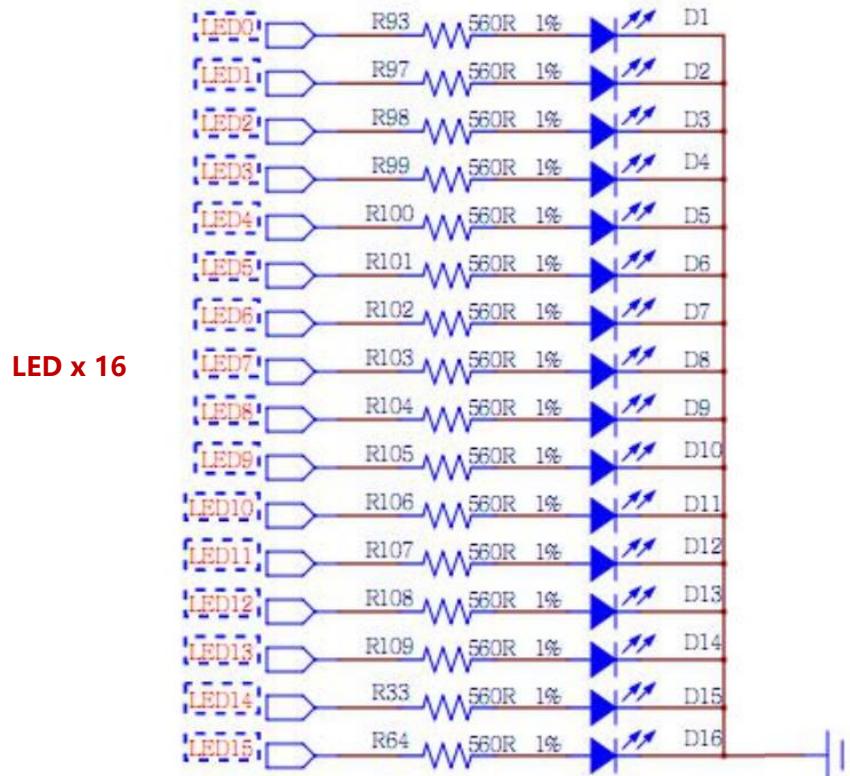
LD1_4->J5, LD1_5->H5, LD1_6->H6, LD1_7->K1

LD2_0->K2, LD2_1->J2, LD2_2->J3, LD2_3->H4

LD2_4->J4, LD2_5->G3, LD2_6->G4, LD2_7->F6

- FPGA管脚高电平点亮

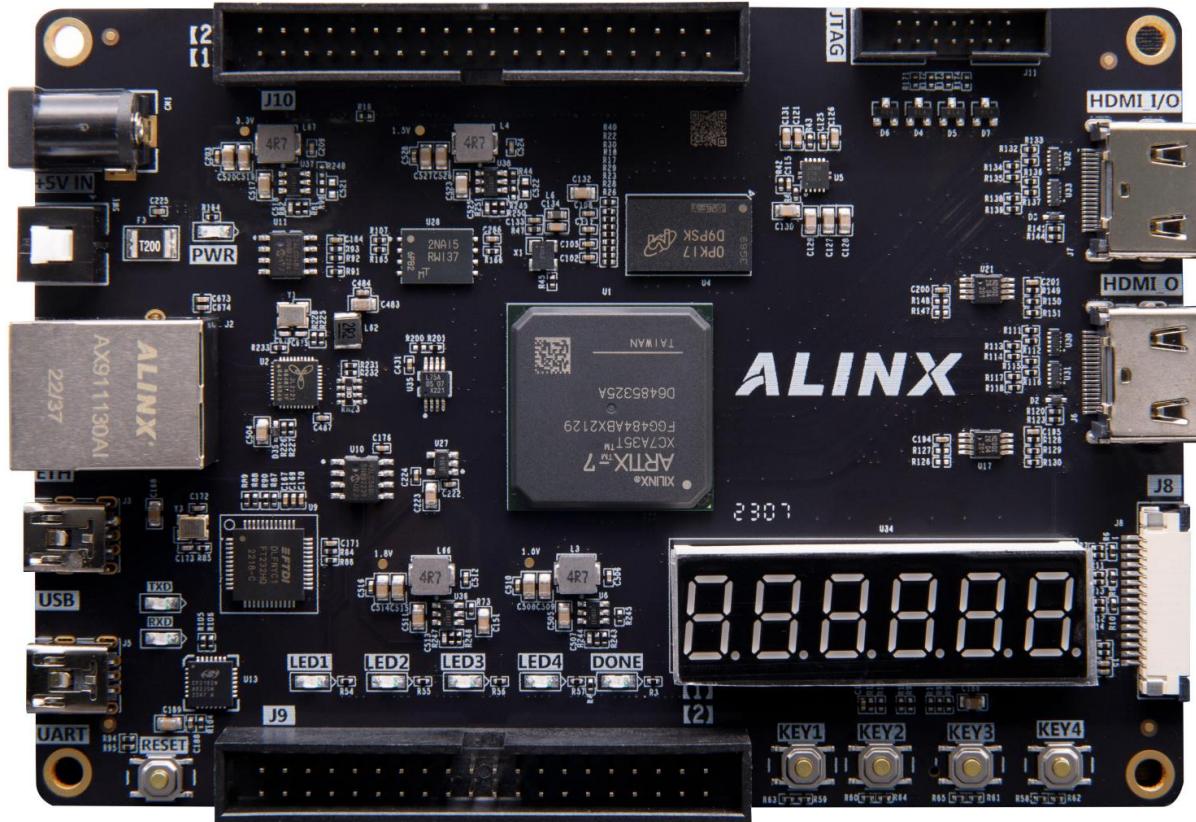
EOG1 LED资源



其他类型开发板介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

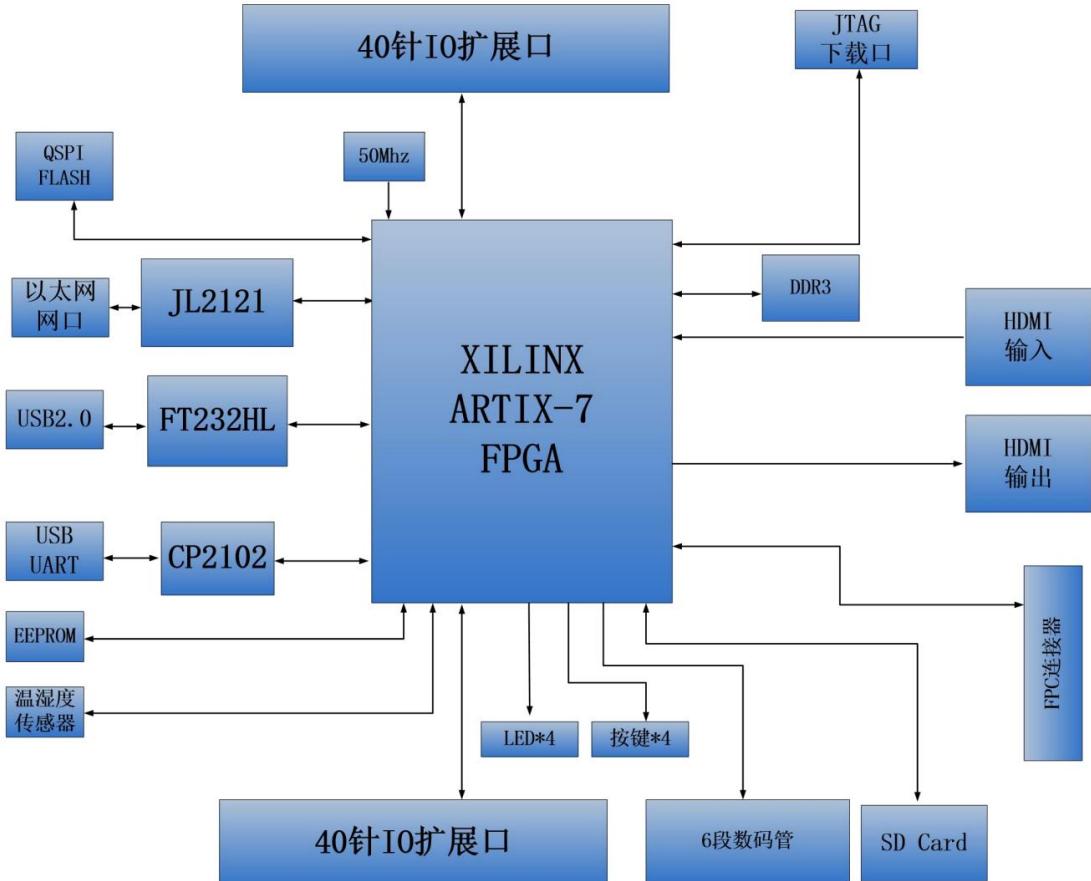


其他类型开发板介绍



■ 黑金FPGA开发板

- XC7A35T-2FGG484I
- 256MB DDR3
- HDMI输入、HDMI输出
- 千兆以太网
- USB2.0
- 两路 40 针扩展口

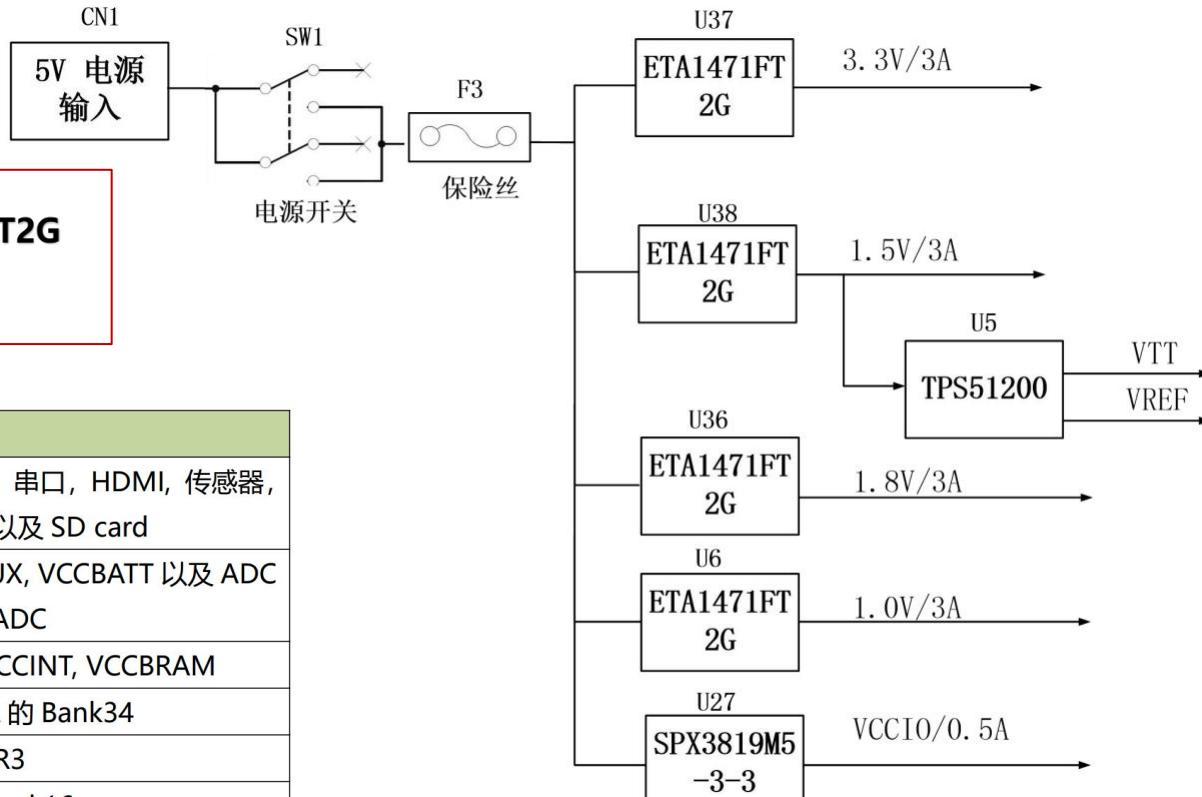


其他类型开发板介绍



- 四路 DC/DC 电源芯片 ETA1471FT2G
- 有上电时序控制

电源	功能
+3.3V	FPGA 的 VCCIO, 以太网, 串口, HDMI, 传感器, FLASH, EEPROM 以及 SD card
+1.8V	FPGA 的辅助电压 VCCAUX, VCCBATT 以及 ADC 电源 VCCADC
+1.0V	FPGA 的核心电压 VCCINT, VCCBRAM
+1.5V	DDR3, FPGA 的 Bank34
VREF, VTT	DDR3
VCCIO	FPGA Bank16



其他类型开发板介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

开发板上提供一个 Sitime 公司的 50M 有源晶振给 FPGA 作为系统时钟输入。晶振输出连接到 FPGA 的全局时钟(GCLK Pin Y18)，这个 GCLK 可以用来驱动 FPGA 内的用户逻辑电路，用户可以通过配置 FPGA 内部的 PLL 和 MMCM 来实现更高的时钟。

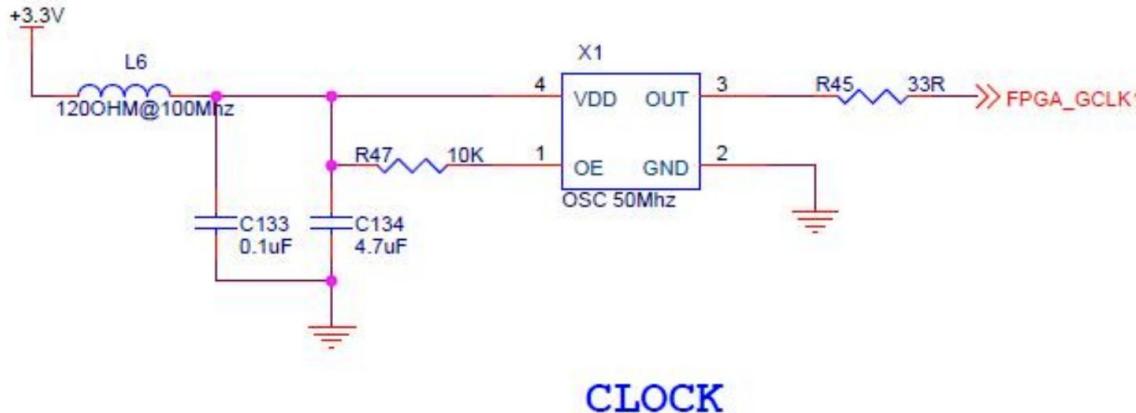


图 5-1 50M 有源晶振

时钟引脚分配：

引脚名称	FPGA 引脚
FPGA_GCLK1	Y18

其他类型开发板介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

AX7035B 板上配有一个 Micron(美光) 的 2Gbit (256MB) 的 DDR3 芯片,型号为 MT41J128M16HA-125。DDR 的总线宽度共为 16bit。DDR3 SDRAM 的最高运行时钟速度可达 400MHz(数据速率 800Mbps)。该 DDR3 存储系统直接连接到了 FPGA 的 BANK 34 的存储器接口上。DDR3 SDRAM 的具体配置如下表 6-1 所示。

表 6-1 DDR3 SDRAM 配置

位号	芯片类型	容量	厂家
U4	MT41J128M16HA-125	128M x 16bit	micron

DDR3 电路

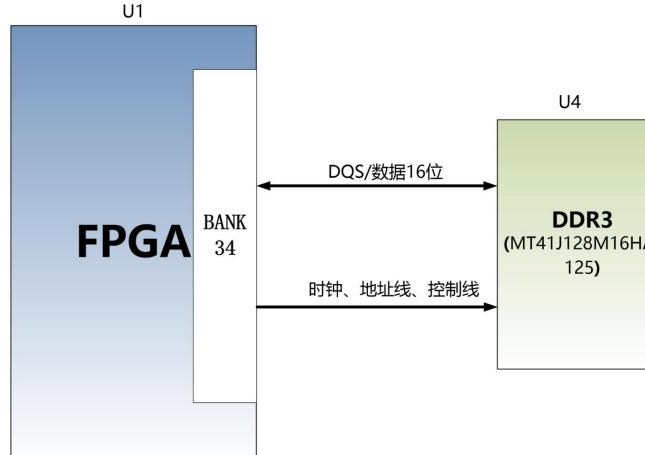


图6-1 DDR3 DRAM原理图示意图

其他类型开发板介绍



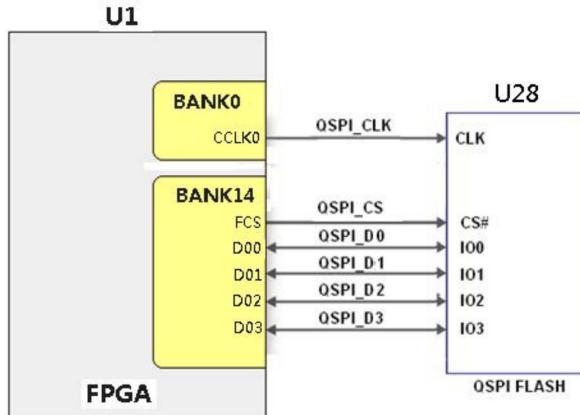
开发板上使用了一片 128Mbit 大小的 QSPI FLASH 芯片，型号为 N25Q128，它使用 3.3V CMOS 电压标准。由于它的非易失特性，在使用中，QSPI FLASH 可以作为 FPGA 系统的启动镜像。这些镜像主要包括 FPGA 的 bit 文件、软核的应用程序代码以及其他的数据文件。

QSPI FLASH的具体型号和相关参数见下表

位号	芯片类型	容量	厂家
U8	N25Q128	128M Bit	Numonyx

表7-1 QSPI Flash的型号和参数

QSPI FLASH



配置芯片引脚分配：

信号名称	FPGA 引脚名	FPGA 管脚号
QSPI_CLK	CCLK_0	L12
QSPI_CS	IO_L6P_T0_FCS_B_14	T19
QSPI_DQ0	IO_L1P_T0_D00_MOSI_14	P22
QSPI_DQ1	IO_L1N_T0_D01_DIN_14	R22
QSPI_DQ2	IO_L2P_T0_D02_14	P21
QSPI_DQ3	IO_L2N_T0_D03_14	R21

图 7-1 QSPI Flash连接示意图

其他类型开发板介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

千兆网口

AX7035B 开发板上通过一片景略半导体的 JL2121-N040I 以太网 PHY 芯片为用户提供网络通信服务。以太网 PHY 芯片是连接到 ARTIX7 FPGA 的 IO 接口上。JL2121-N040I 芯片支持 10/100/1000 Mbps 网络传输速率, 通过 RGMII 接口跟 FPGA 进行数据通信。KSZ9031RNX 支持MDI/MDX 自适应, 各种速度自适应, Master/Slave 自适应, 支持 MDIO 总线进行 PHY 的寄存器管理。

以太网 PHY 的 FPGA 引脚分配如下:

信号名称	FPGA 引脚号	备注
E1_GTXC	L14	RGMII 发送时钟
E1_TXD0	J21	发送数据 bit0
E1_TXD1	M20	发送数据 bit1
E1_TXD2	L18	发送数据 bit2
E1_TXD3	L20	发送数据 bit3
E1_TXEN	L19	发送使能信号
E1_RXC	K18	RGMII 接收时钟
E1_RXD0	K19	接收数据 Bit0
E1_RXD1	M15	接收数据 Bit1
E1_RXD2	J17	接收数据 Bit2
E1_RXD3	J20	接收数据 Bit3
E1_RXDV	M21	接收数据有效信号
E1_MDC	K17	MDIO 管理时钟
E1_MDIO	K16	MDIO 管理数据
E1_RESET	L15	PHY 芯片复位

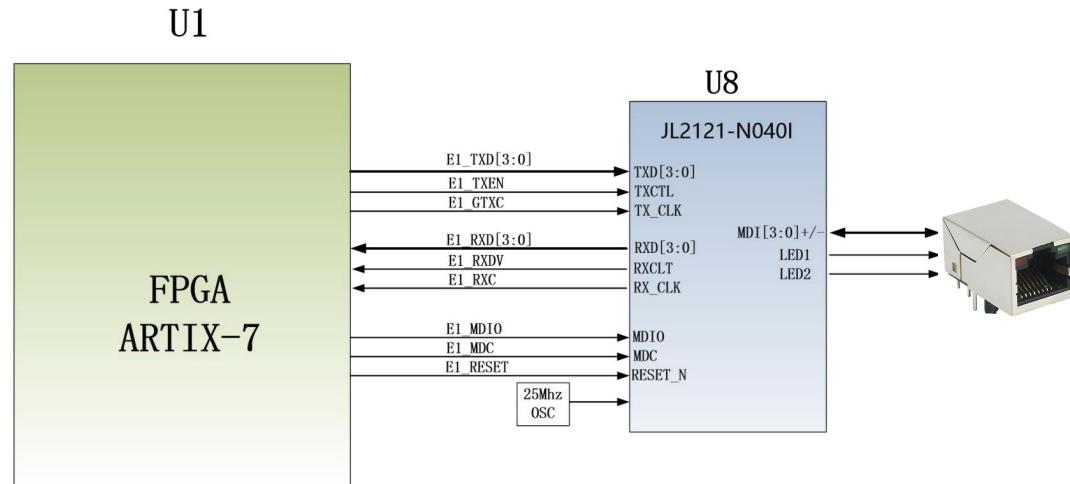


图 8-1 FPGA 与 PHY 芯片连接示意图

其他类型开发板介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

AX7035B 开发板上 HDMI 输出接口的实现是通过 FPGA 的差分 IO 直接连接到 HDMI 接口的差分信号和时钟，在 FPGA 内部实现数据进行编码和并行转差分转换后对 HDMI 信号的差分输出，实现 HDMI 数字视频输出的传输解决方案，最高支持 1080P@60Hz 输出的功能。

HDMI 的差分驱动信号通过 FPGA 的 BANK35 上 IO 输出，在信号接口处我们加了 ESD 保护器件，另外 HPD(hot plug detect)信号作为用来检测外部 HDMI 显示设备是否插入，图 9-1 为 HDMI 输出设计的原理图。

HDMI输出

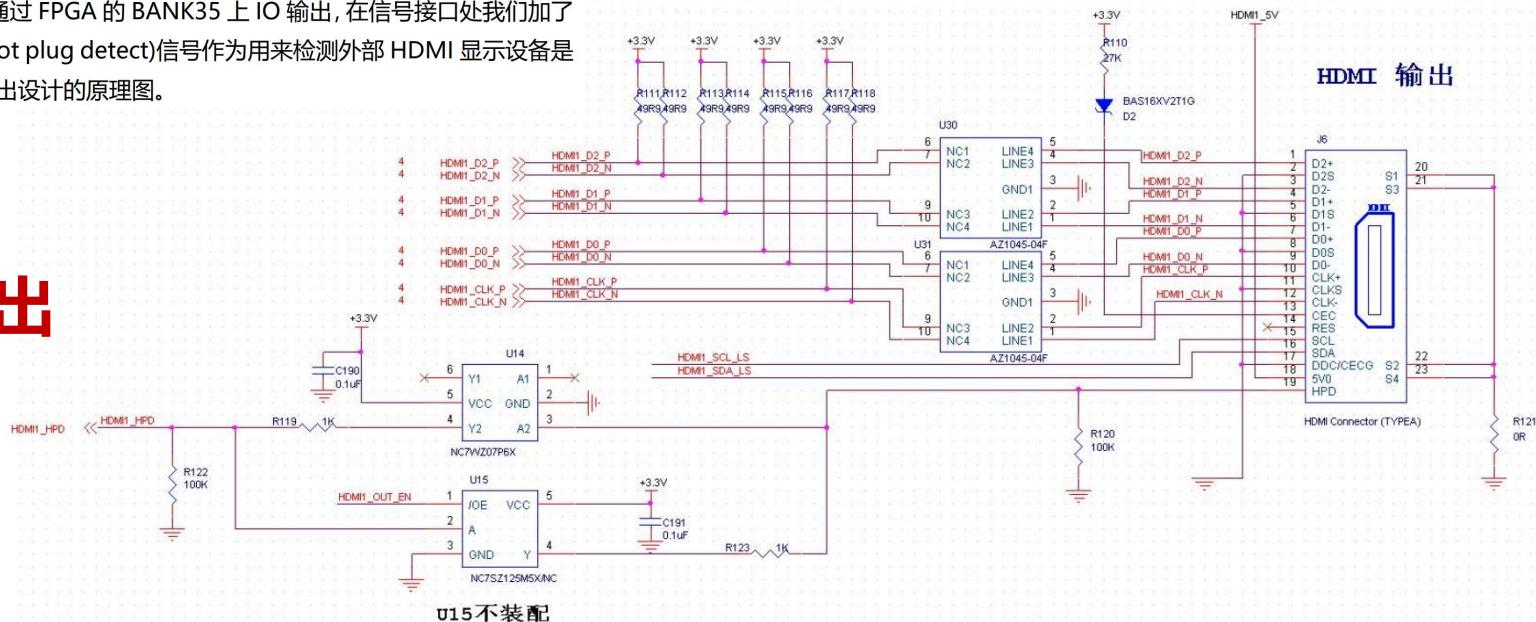


图 9-1 HDMI 输出接口原理图

其他类型开发板介绍



AX7035B 采用了 FTDI Chip 公司的 FT232H 单通道高速 USB 芯片为开发板实现和电脑之间的 USB2.0 数据通信。最高 USB2.0 高速通信 (480Mb/s) 和全速通信 (12Mb/s) , 数据接口支持不同的数据通信模式 (FIFO, I2C, SPI, JTAG) , 上电后读取外置的 EEPROM 配置内容来决定数据通信模式, 也可以通过 PC 方便的修改配置方式。USB 芯片的接口管脚的功能是复用的, 具体请参考 FT232H 的芯片手册。

USB2.0

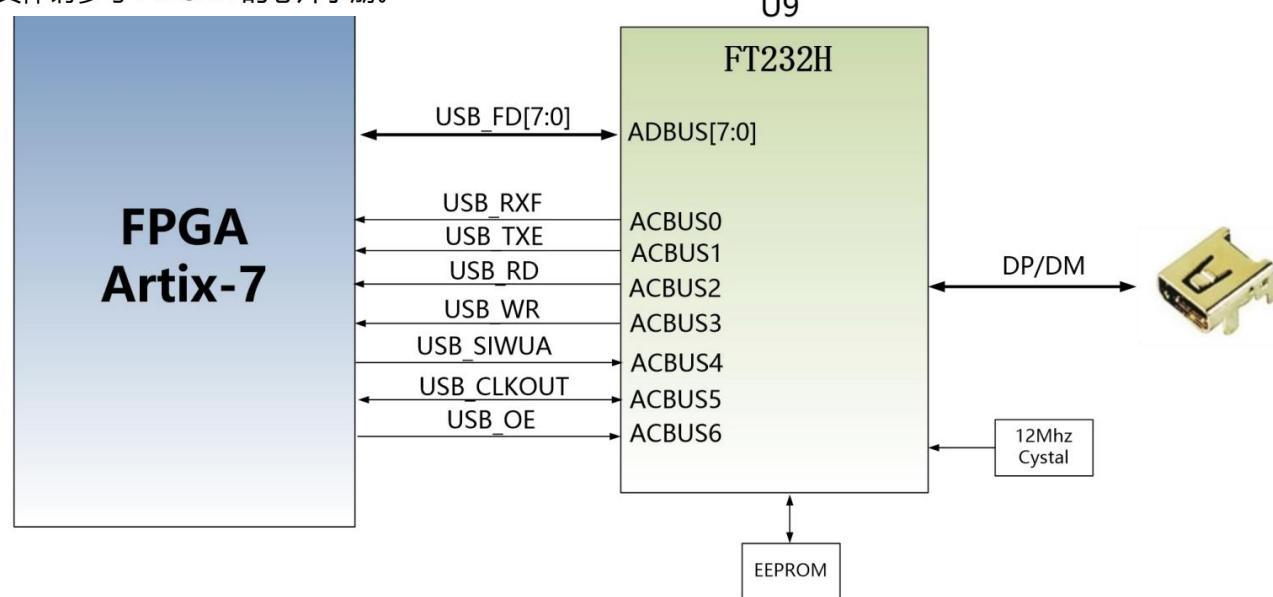


图 11-1 USB2.0 接口原理图

其他类型开发板介绍



SD 卡(Secure Digital Memory Card)是一种基于半导体闪存工艺的存储卡，1999年由日本松下主导概念，参与者东芝和美国 SanDisk 公司进行实质研发而完成。2000年这几家公司发起成立了 SD 协会(Secure Digital Association 简称 SDA)，阵容强大，吸引了大量厂商参加。其中包括 IBM, Microsoft, Motorola, NEC、Samsung 等。在这些领导厂商的推动下，SD 卡已成为目前消费数码设备中应用最广泛的一种存储卡。

SD 卡是现在非常常用的存储设备，我们扩展出来的 SD 卡，支持 **SD 和 SPI 模式**，使用的 SD 卡为 MicroSD 卡。原理图如下图 12-1 所示。

SD卡

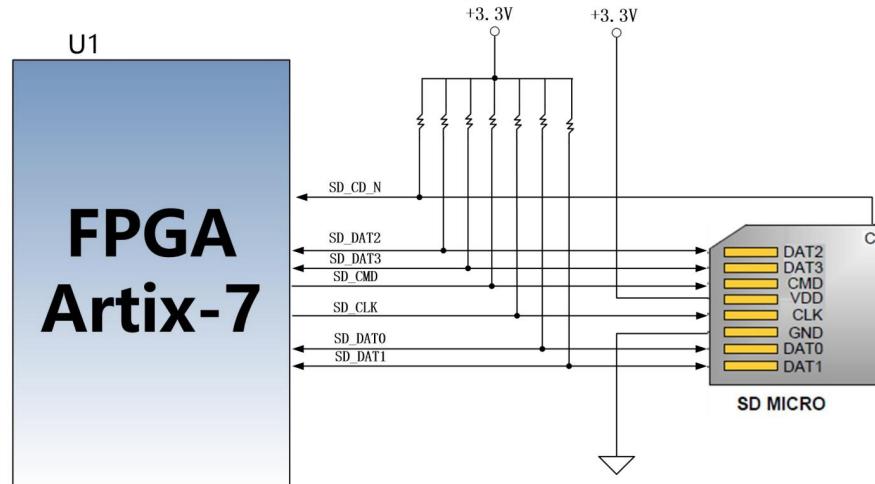
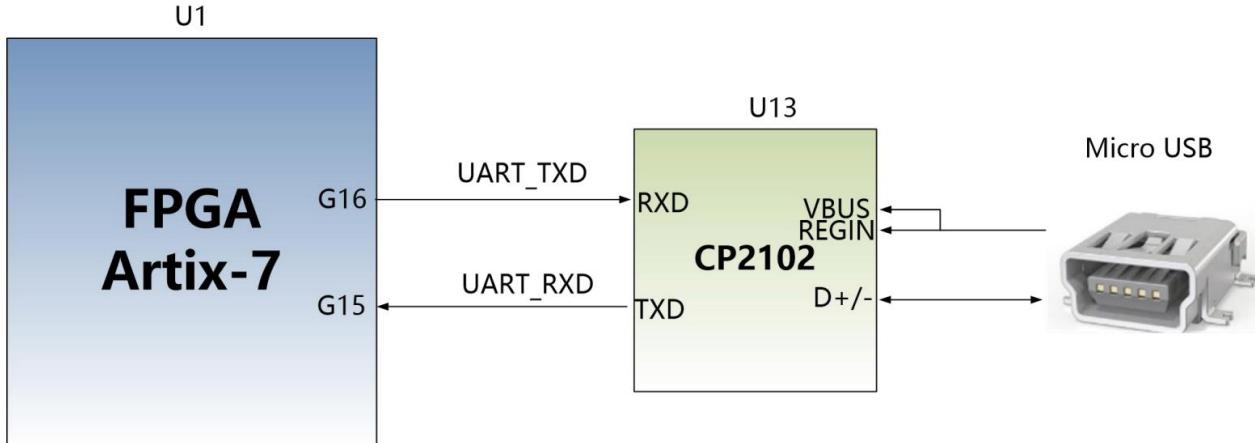


图 12-1 SD 卡槽原理图

其他类型开发板介绍



AX7035B 开发板包含了 Silicon Labs CP2102GM 的 USB-UAR 芯片, USB 接口采用 MINI USB 接口,可以用一根 USB 线将它连接到上 PC 的 USB 口进行串口数据通信 。
USB Uart 电路设计的示意图如下图所示:



USB转串口

13-1 USB 转串口示意图

其他类型开发板介绍



AX7035B 开发板板载了一片 EEPROM，型号为 24LC04, 容量为： 4Kbit (2*256*8bit) , 由 2 个 256byte 的 block 组成, 通过 IIC 总线进行通信。板载 EEPROM 就是为了学习 IIC 总线的通信方式。EEPROM 的 I2C 信号连接的 FPGA 的 IO 口上。下图 14-1 为 EEPROM 的设计示意图

EEPROM

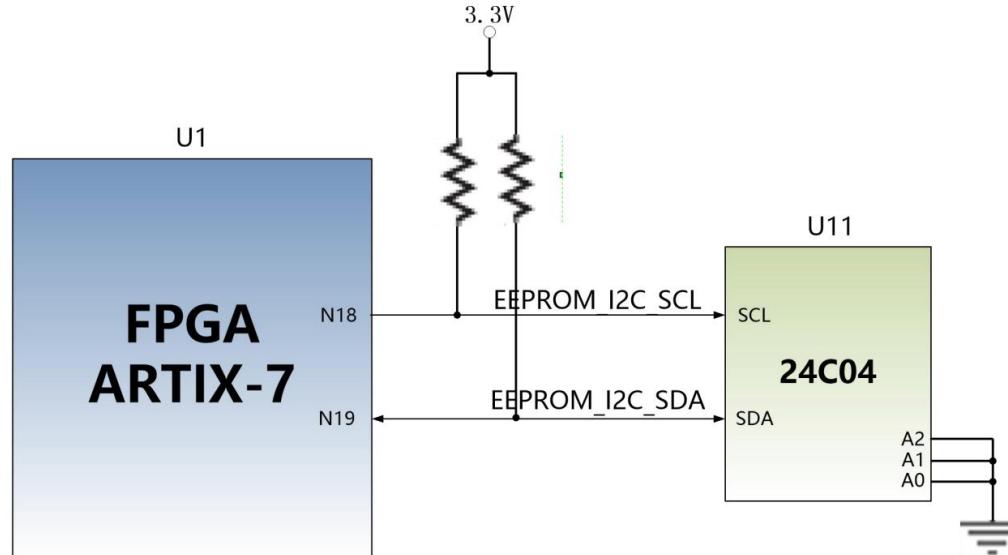


图 14-1 EEPROM 原理图部分



THANKS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第三讲 (3.2 HDL语言概述)

2025年3月18日



为什么FPGA难学？

- ◆ 不熟悉FPGA的内部结构，CLB、触发器、布线资源、BRAM、CMT、IOB... ...
- ◆ 错误理解HDL语言，不理解HDL描述的硬件结构，HDL语言的本质是在描述硬件
- ◆ 数字电路知识薄弱，尤其是不理解时钟对时序电路的重要作用（时钟是时序电路的掌控者）

如何学？

- ◆ 必须熟练掌握一门HDL语言，多看例程代码，多实践练习
- ◆ 独立完成中小规模的数字系统设计，Lab1.....Lab9.....
- ◆ 理解设计方法和设计原则（面积与速度平衡、乒乓操作、流水线、跨时钟域...），学会如何提高开发效率
- ◆ 增强理论知识（数字电路、信号处理、硬件接口、CPU、控制理论、通信理论...）

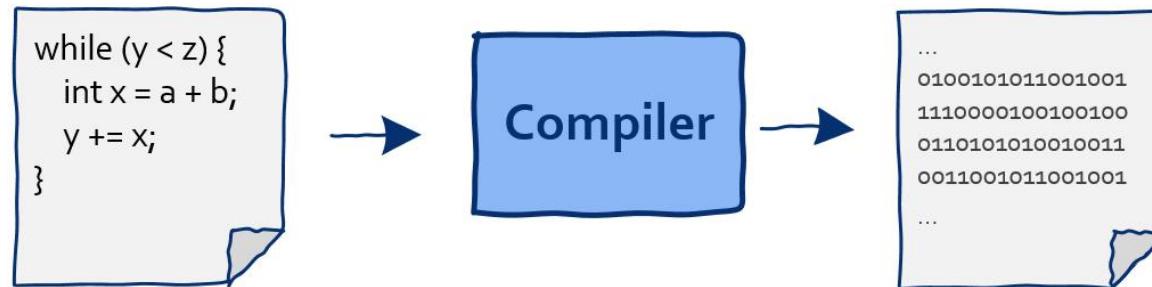


玄学？



编译器的本质？

- ◆ 编译器是一做**桥梁**，是一个**译者**，编译器将**人类的语言**（想法）**翻译成机器语言**
- ◆ FPGA编译器将HDL语言综合、实现、生成bit文件，最终变成机器语言（CLB、布局布线、IOB、CMT等实现形式）
- ◆ 硬件功能与预想不一致的主要原因：（1）想错了（设计错误，代码错误）；（2）编译器听不懂；（3）编译器做不到



核心问题是描述正确的同时能够让编译器理解



◆ HDL语言简介

- ◆ HDL语言风格
- ◆ HDL语言与C语言
- ◆ VHDL与Verilog语言
- ◆ HDL的学习

◆ HDL语言与硬件实现

- ◆ HDL代码与电路的映射关系

◆ 可综合的Verilog HDL语法

- ◆ 模块声明语法
- ◆ 端口声明语法
- ◆ 参数定义
- ◆ 信号类型
- ◆ 多语句定义
- ◆ 判断语法
- ◆ 循环语句
- ◆ 任务定义
- ◆ 连续赋值
- ◆ always模块
- ◆ 运算操作符
- ◆ 赋值符号 (阻塞赋值与非阻塞赋值)

HDL语言风格



四种描述风格

- ◆ 数据流风格：使用连续赋值语句
- ◆ 行为风格：使用initial、always语句
- ◆ 结构风格：模块例化（类似于C++类的例化）、原语
- ◆ 混合风格：上述风格的混合，实际中最常用

```
always@(posedge clk or negedge rst)begin
  if(!rst)
    bps_DR <= 16'd10414;
  else begin
    case(Baud_Set)
      0:bps_DR <= 16'd10414;
      1:bps_DR <= 16'd5207;
      2:bps_DR <= 16'd2603;
      3:bps_DR <= 16'd1301;
      4:bps_DR <= 16'd876;
      default:bps_DR <= 16'd10414;
    endcase
  end
end

always @(*) begin
  case (cur_state_tx)//full case
    S0 : begin //判断FIFO中有数据，跳转S1
      if (empty == 0) begin
        next_state_tx = S1;
      end
      else next_state_tx = S0;
    end
    S1 : begin //rd_en信号输出，直接跳转
      next_state_tx = S2;
    end
    S2 : begin //从FIFO获取数据，直接跳转
      next_state_tx = S3;
    end
    S3 : begin //跟随波特率节拍输出数据，本状态高电平
      if (bps_clk == 1) begin
        next_state_tx = S4;
      end
      else
        next_state_tx = S3;
    end
  end
end
```

行为风格

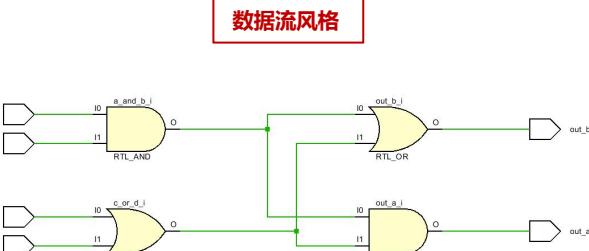
```
module gates(
  input a,
  input b,
  input c,
  input d,
  output out_a,
  output out_b
);

wire a_and_b;
wire c_or_d;

assign a_and_b = a & b;
assign c_or_d = c | d;
assign out_a = a_and_b & c_or_d;
assign out_b = a_and_b | c_or_d;

endmodule
```

数据流风格



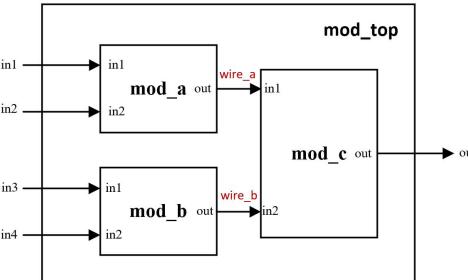
```
module mod_top(
  input in1,
  input in2,
  input in3,
  input in4,
  output out
);

wire wire_a;
wire wire_b;

mod_a instance1 (.in1(in1),.in2(in2),.out(wire_a));
mod_b instance2 (.in1(in3),.in2(in4),.out(wire_b));
mod_c instance3 (.in1(wire_a),.in2(wire_b),.out(out));

endmodule
```

结构风格





HDL语言为硬件描述语言 (Hardware Description Language)

- 与高级程序语言（C）类似，HDL是一种高级程序设计语言，但与C语言的设计思想完全不同
- 软件高级程序语句是对通用型处理器（如CPU）的编程，主要是在固定硬件体系结构下的软件化程序设计。处理器的体系结构和功能决定了可以用于程序编程的固定指令集，设计人员的工作是调用这些指令，在固化的体系结构下实现特定的功能。
- Verilog HDL和VHDL等硬件描述语言是对电路的设计，将基本的最小数字电路的单元（如门单元、寄存器、存储器等）通过连接方式构成具有特定功能的硬件电路。
- HDL程序设计的正确性需要通过综合后电路的正确性来验证，逻辑上相同其物理电路却有可能完全不同

```
int main()
{
    int a = 1;
    int b = 2;
    a = a+b;
}
```

编译

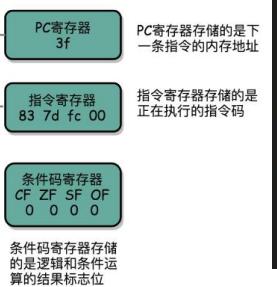
```
push rbp
mov rbp, rsp
mov DWORD PTR [rbp-0x4], 0x1
mov DWORD PTR [rbp-0x8], 0x2
mov eax, DWORD PTR [rbp-0x8]
Add DWORD PTR [rbp-0x4], eax
pop rbp
ret
```

汇编

```
0: 55
1: 48 89 e5
4: c7 45 fc 01 00 00 00
b: c7 45 f8 02 00 00 00
12: b8 45 f8
15: 01 45 fc
18: 5d
19: c3
```

内存地址	程序	指令代码
3b	指令1	83 7d fc 00
3f	指令2	75 09
41	指令3	c7 45 f8 01 00
	
4a	指令N	c7 45 f8 02 00
51	指令N+1	b8 00 00 00 00

计算机执行C语言的方式



体会一下HDL语言与C语言的不同

Verilog HDL设计8位计数器

```
51 module counter(clk,rst_n,cnt
52     input clk,rst_n;
53     output[7:0] cnt;
54 );
55     reg [7:0] cnt;
56     always @(posedge clk or negedge rst_n)
57         if (!rst_n)cnt<=8'h00000000;
58         else cnt<=cnt+1'b1;
59 endmodule
```

下面代码套用了C语言的思路

```
60 module counter(clk,rst_n,cnt
61     input clk,rst_n;
62     output[7:0] cnt;
63 );
64     reg [7:0] cnt;
65     integer i;
66     always @(posedge clk or negedge rst_n) begin
67         if (!rst_n)cnt<=8'h00000000;
68         else
69             for (i = 0 ;i<=255 ;i=i+1 ) begin
70                 cnt<=cnt+1'b1;
71             end
72 endmodule
```

VHDL与Verilog语言



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

VHDL与Verilog

异		同
Verilog	VHDL	
与C语言类似，在C语言的基础上，增加一些特殊的约定，易于掌握和学习	“异类”语言，几乎没有语法、格式类似的编程语言，相对难上手，不直观。	Verilog HDL和VHDL都是用于逻辑设计的硬件描述语言，并且都已成为IEEE标准，成为公认行业标准
1983年诞生于Gateway Automation公司，1993年全面应用于几乎所有的ASIC以及FPGA厂商，经三次修改，发布IEEE1364-2005标准	20世纪80年代初期诞生于美国军方，1987年被IEEE和没干过国防部确认为标准硬件描述语言，最新标准为IEEE1076-1993	能形式化地抽象表示电路的行为和结构、支持逻辑设计中层次与范围的描述、
系统级建模方面较差,发展了SystemVerilog	行为级建模覆盖范围大	可借用高级语言的精巧结构来简化电路行为的描述
通常不进行说明，或只进行非常简短的说明，程序比较简短	需要进行大量说明，程序通常比较长	支持电路描述由高层到低层的综合转换
只有常量和变量，类型种类少 运算时所受的约束少	常量、信号、变量共9种预定义类型，必须进行类型说明，运算时必须考虑类型的一致性和适用性	具有电路仿真与验证机制以保证设计的正确性
运算划分比较具体，对逻辑代数反映更细致	运算划分比较抽象，适应面较广	硬件描述与实现工艺无关



◆ HDL语言学习方法

- ◆ 建议同时掌握VHDL和Verilog
 - ◆ 建议初学者先掌握其中一门，至于到底先下手哪一门，则需要根据自身的情况做考量
 - ◆ 也别忘了兼顾另一门，无论哪一种语言，要能看懂别人的设计
- ◆ 代码和电路的关系
 - ◆ 不要抛开实际电路而研究语法，多花些精力比对实际逻辑电路和代码的映射关系，必要时做做仿真，最好能再找些直观的外设在实验板上看看结果
 - ◆ 若能达到代码和电路都心中有数，那才证明真真正正掌握HDL语言的精髓了



◆ HDL语言简介

- ◆ HDL语言与C语言
- ◆ VHDL与Verilog语言
- ◆ HDL的学习

◆ HDL语言与硬件实现

- ◆ HDL代码与电路的映射关系

◆ 可综合的Verilog HDL语法

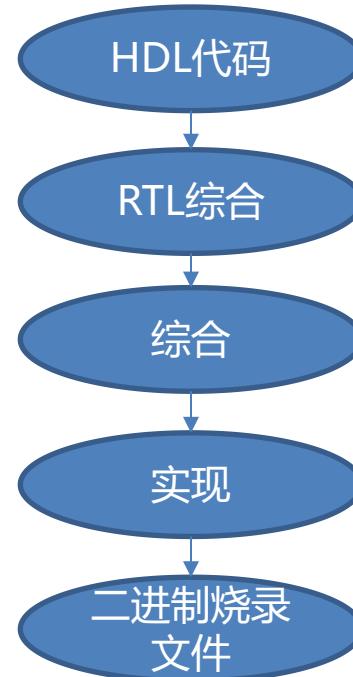
- ◆ 模块声明语法
- ◆ 端口声明语法
- ◆ 参数定义
- ◆ 信号类型
- ◆ 多语句定义
- ◆ 判断语法
- ◆ 循环语句
- ◆ 任务定义
- ◆ 连续赋值
- ◆ always模块
- ◆ 运算操作符
- ◆ 赋值符号 (阻塞赋值与非阻塞赋值)



◆ HDL代码与电路的映射关系

◆ 软件流程

- ◆ 编写“HDL”描述电路的功能
- ◆ 在EDA工具中进行“RTL综合”和“综合”
- ◆ “实现” 将对FPGA器件进行最终的布局布线
- ◆ 最后转化可以在FPGA上运行的二进制烧录文件





◆ HDL代码与电路的映射关系

◆ 1.HDL代码

module 模块名 (接口名)

 定义输入输出

 描述电路功能

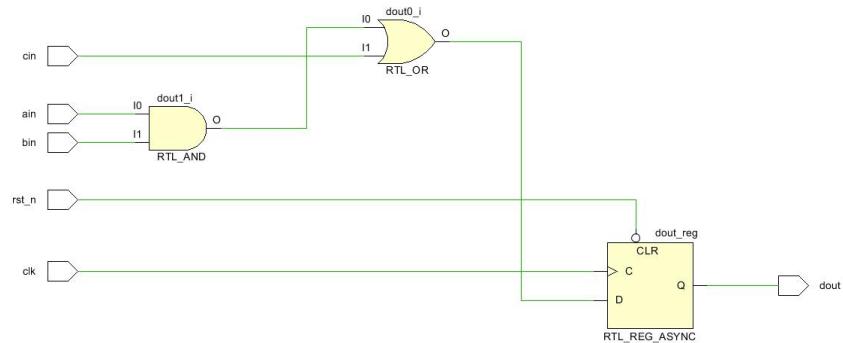
◆ 2.RTL综合

◆ 将HDL代码转化为逻辑门电路的形式来表达

◆ Vivado中的RTL Analysis可以将代码进行逻辑性的解释

◆ 生成的电路图与代码设计本身完全一致

```
1 module at7();
2     clk, rst_n,
3     ain, bin, cin, dout
4 );
5     input clk;
6     input rst_n;
7     input ain,bin,cin;
8     output reg dout;
9     always @(posedge clk or negedge rst_n)
10        if(!rst_n) dout<= 1'b0;
11        else dout <= (ain&bin) | cin;
12 endmodule
```



HDL语言与硬件实现

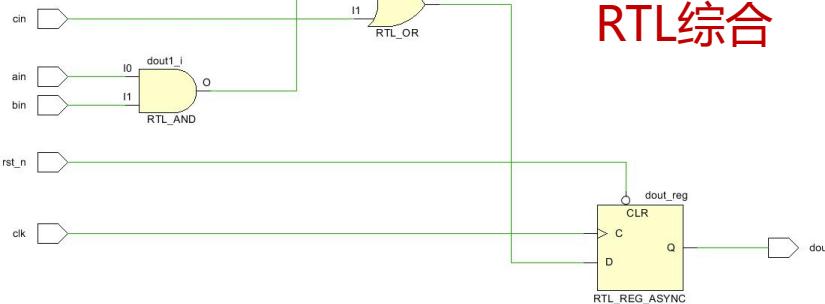


國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

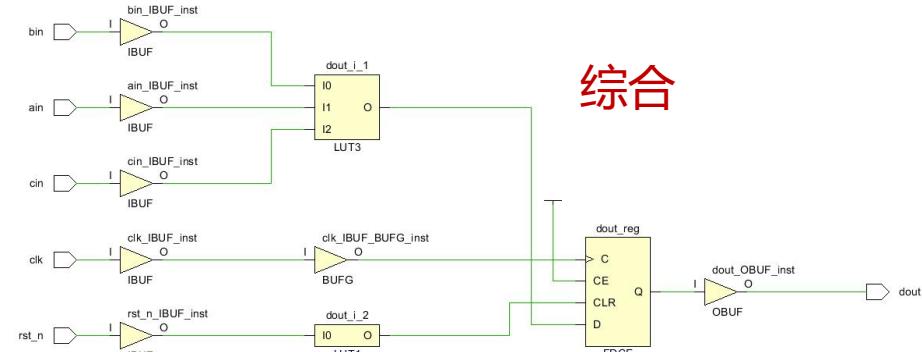
◆ HDL代码与电路的映射关系

◆ 3.综合

◆ 综合与RTL综合不同，“综合”是将设计进一步转化为
FPGA物理结构相对应的电路形式



RTL综合



综合



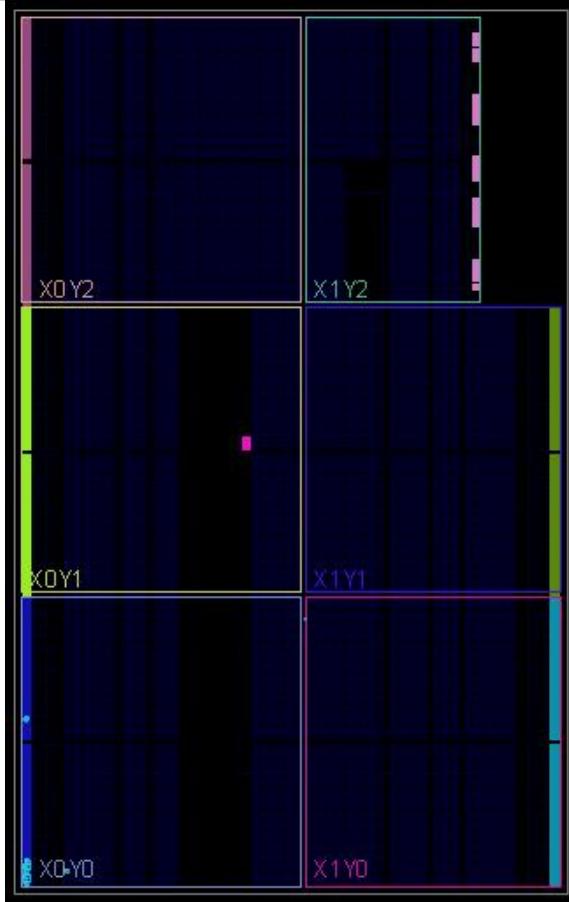
◆ HDL代码与电路的映射关系

◆ 4. 实现

- ◆ 在“综合”给出HDL代码与FPGA器件的映射关系后，做详细部署的工作
- ◆ 相当于PCB设计中，“综合”是“原理图阶段”，“实现”是“布局布线（layout）”阶段

◆ 5. 生成烧录文件

- ◆ 完成上述工作后，生成一个bit文件用于烧录到FPGA中运行





◆ HDL语言简介

- ◆ HDL语言与C语言
- ◆ VHDL与Verilog语言
- ◆ HDL的学习

◆ HDL语言与硬件实现

- ◆ HDL代码与电路的映射关系

◆ 可综合的Verilog HDL语法

- ◆ 模块声明语法
- ◆ 端口声明语法
- ◆ 参数定义
- ◆ 信号类型
- ◆ 多语句定义
- ◆ 判断语法
- ◆ 循环语句
- ◆ 任务定义
- ◆ 连续赋值
- ◆ always模块
- ◆ 运算操作符
- ◆ 赋值符号 (阻塞赋值与非阻塞赋值)



Verilog基础语法

- ◆ 可综合语法是指硬件能够实现的Verilog语法，是Verilog语法中很小的一部分子集
- ◆ Verilog语句区分大小写
- ◆ 格式自由，可以在一行内编写，也可跨多行编写(不推荐一行内编写方式)
- ◆ 每个语句必须以分号为结束符
- ◆ 空白符号（换行、制表、空格）没有实际意义，编译时忽略
- ◆ 两种注释方法：// 单行注释； /* */跨行注释
- ◆ 标识符：任意一组字母、数字、\$ 符号和 _ (下划线)符号的组合，第一个字符必须是字母或者下划线
- ◆ 关键字：Verilog 中预留的用于定义语言结构的特殊标识符，全部为小写字符， reg, wire, always, if else, input... ...

可综合的Verilog HDL语法



1. 模块声明语法：module...endmodule

- ◆ 在每个Verilog文件中都会出现该语法
- ◆ 推荐Verilog-2001的用法
- ◆ Module后的name为该module的命名，默认数字、下划线和字母的组合均可，第一个字符必须是字母或者下划线
- ◆ 随后一个()内罗列出该模块所有的输入输出端口信号名
- ◆ 接着一直到endmodule之间就是要实现功能的逻辑代码
- ◆ module是对数字电路的数据、功能和时序的封装
- ◆ 实际意义是代表硬件电路上的逻辑实体

```
1 module RFSP_model (
2   Q,
3   CLK,
4   CEN,
5   WEN,
6   A,
7   D
8 );
9 output [32-1:0] Q;
10 input CLK;
11 input CEN;
12 input WEN;
13 input [32-1:0] A;
14 input [32-1:0] D;
15 reg [32-1:0] Q;
16 ...
17 endmodule
```

verilog-1995

```
1 module RFSP_model (
2   output reg [32-1:0] Q,
3   input CLK,
4   input CEN,
5   input WEN,
6   input [32-1:0] A,
7   input [32-1:0] D
8 );
9 ...
10 endmodule
```

verilog-2001

```
14 module name(<端口信号列表>...);
15 /*端口定义
16 /*数据类型说明
17 /*参数说明
18 /*<逻辑代码>... */
19 endmodule
```

可综合的Verilog HDL语法



2. 端口声明：input, output, inout

- ◆ inout用法比较特殊，不常用
- ◆ 每个module都会有输入输出的信号用于和外部器件或其他module进行数据交互
- ◆ 对于本地module而言，这些信号无非可以归为三类，即输入（input）信号，输出（output）信号和双向（inout）信号
- ◆ 这些端口信号名都要在module名后的“（）”内列出

3. 参数定义：parameter

- ◆ parameter用于申明一些常量，主要是便于模块的移植或升级时的修改
- ◆ 在功能上类似C语言的 #define
- ◆ 对于一个可读性强的代码来说，parameter是必不可少的

```
22 module name(  
23   input clk,  
24   input rst,  
25   output [7:0] led);  
26   /*<逻辑代码>...*/  
27 endmodule
```

```
//输出的数据  
parameter NUM1 = 8'hAA;  
parameter NUM2 = 8'hBB;  
parameter NUM3 = 8'hDD;  
parameter NUM4 = 8'hEE;  
parameter NUM5 = 8'hFF;  
parameter NUM6 = 8'h11;  
parameter NUM7 = 8'h22;  
parameter NUM8 = 8'h33;
```

可综合的Verilog HDL语法



4.信号类型：wire, reg等

- ◆ 作为input或inout的信号端口只能是wire型，而output则可以是wire也可以是reg
- ◆ 虽然在代码中我们可以定义信号为wire或reg类型，但是实际的电路实现是否和我们预先的一致还要看综合工具的表现

5.多语句定义：begin...end

- ◆ 通俗的讲，他就是C语言中的“{}”
- ◆ 用于单个语法的多个语句定义
- ◆ 例如if判断正确，要执行多个语句时，需要使用begin...end，执行省略号中的语句

```
29 module <name>(<端口命名>...);  
30 //输入端口申明  
31 input <端口命名1>;  
32 input wire <端口命名2>;  
33 input [<最高位>:<最低位>] <端口命名3>;  
34 ...  
35 //输出端口申明  
36 output <端口命名4>;  
37 output [<最高位>:<最低位>] <端口命名5>;  
38 output reg [<最高位>:<最低位>] <端口命名6>;  
39 ...  
40 //参数定义  
41 Parameter<参数命名1>=<默认值1>;  
42 Parameter[<最高位>:<最低位>]<参数命名2>=<默认值2>;  
43 ...  
44 <逻辑代码>...  
45 endmodule
```

```
49 //含有命名的begin语句  
50 begin: <块名>  
51 | //可选声明部分  
52 | //具体逻辑  
53 end  
54  
55 //基本的begin语句  
56 begin  
57 | //可选声明部分  
58 | //具体逻辑  
59 end
```

可综合的Verilog HDL语法



6. 比较判断：if...else, case...default...endcase

- ◆ If else以及case是最常用的语法，与C语言基本一致
- ◆ 只执行一条语句的时候不需要使用begin end

7. 循环语句：for

- ◆ for语句用的比较少，但也会在一些特定的设计中使用

```
81  for (<变量名> =<初值> ; <判断表达式>; <变量名>=<新值>)
82    begin
83      //具体逻辑
84    end
```

```
61  if(<判断条件1>)
62    begin
63      //具体逻辑1
64    end
65  else if(<判断条件2>)
66    begin
67      //具体逻辑2
68    end
69  else
70    begin
71      //具体逻辑3
72    end
73
74  case(<判断变量>)
75    <取值1>: <具体逻辑1>
76    <取值2>: <具体逻辑2>
77    <取值3>: <具体逻辑3>
78    default:<具体逻辑4>
79  endcase
```

可综合的Verilog HDL语法



8.任务定义: task...endtask

- ◆ 类似于C语言中的子函数
- ◆ 其中可以有input、output和inout端口作为参数
- ◆ 没有返回值，所以不可用于表达式中

9.连续赋值: assign,问号表达式 (? :)

- ◆ assign用于直接互联不同的信号或直接给wire变量赋值
- ◆ ?:表达式就是简单的if...else语句，更多的使用在**组合逻辑中**
- ◆ ?:表达式一般需要结合特定的语境使用

```
87  task <task命名>;  
88      //可选声明部分  
89      begin  
90          //具体逻辑  
91      end  
92  endtask
```

```
assign <wire变量名> = <变量或常量>;
```

```
assign <wire变量名> = (判断条件)?(判断条件为真时的逻辑处理):(判断条件为假时的逻辑处理)
```

```
assign out = (a>b?)1'b1:1'b0;
```

可综合的Verilog HDL语法



10.always模块：敏感表可以为电平、沿信号posedge/negedge

- ◆ 通常和@连用
- ◆ always后若有沿信号(上升沿posedge, 下降沿negedge)声明, 则多为时序逻辑

11.运算操作符：各种逻辑操作符、移位操作符、算数操作符

- ◆ Verilog中绝大多数运算操作符是可综合的
- ◆ 有的综合工具是无法综合乘运算和除运算的
- ◆ 加+, 减-, 非!, 取反~, 与&, 与非~&, 或|, 或非~|, 异或^, 同或^~, 取模%, 逻辑左移<<, 逻辑右移>>, 小于<, 小于或等于<=, 大于>, 大于或等于>=, 逻辑相等==, 逻辑不等!=, 逻辑与&&, 逻辑或||

12.赋值符号：=和<=

- ◆ 阻塞赋值：=
- ◆ 非阻塞赋值：<=

```
100 //always在组合逻辑中的使用
101 always@(*)
102 begin
103
104 end
105
106 //单个沿触发的时序逻辑
107 always@(沿变化)
108 begin
109     //具体逻辑
110 end
111 //多个沿触发的时序逻辑
112 always@(<沿变化1>or<沿变化2>)
113 begin
114     //具体逻辑
115 end
```



◆阻塞赋值与非阻塞赋值

- 组合逻辑中使用阻塞赋值
- 阻塞赋值相当于直接连线
- 时序逻辑中使用非阻塞赋值
- 非阻塞赋值相当于经过寄存器



Verilog语法例程1

<https://hdlbits.01xz.net/wiki/Alwaysblock1>

- ◆ 组合逻辑电路的设计实现
- ◆ 可采用assign语句或always@(*)语句，取决于实现的便利性，always块内才可以使用if esle, case等语句
- ◆ 采用always@(*), 避免遗漏敏感信号，导致出现不可预知的问题



◆ 可综合的语法与行为级语法

- ◆ 可综合的语法是一个很小的子集，对于初学者，建议先重点掌握好这个子集
- ◆ 做仿真验证需要用到更高级的语法（不可综合的），后续也须掌握

◆ HDL学习经验

- ◆ 手中需要准备一本比较完整的语法书籍，认真看过、理解过，做到相关语法心中有数
- ◆ 参考一些简单的例程，并且自己动手写写代码实现相同或相近的电路功能
- ◆ 总结下来，就是**多看、多写、多思考、多比对**



Verilog在线学习网站

Verilog Practice

HDLBits is a collection of small circuit design exercises for practicing digital hardware design using Verilog Hardware Description Language. HDLBits problems follow a natural flow, where each problem re-inforces learning challenge your circuit design skills.

Each problem is a small Verilog module that you can copy and paste into a Verilog editor. HDLBits gives you immediate feedback on the circuit module you submit. Your circuit is checked for correctness by comparing with a set of test vectors corresponding to the our reference solution.

How to use HDLBits

1. Choose a problem. [Browse the problems set or go to the first problem.](#)
2. Write a solution in Verilog
3. Submit your solution. If necessary.

If you want to track your progress or move to another browser, create a [username and password](#) so you can log in from elsewhere.

Which exercises should I do?

The exercises are organized by topic and by approximate difficulty within each topic. Start from the "Getting Started" section to get familiar with how to use HDLBits. Then start with the easiest topics, such as basic logic, and work your way up in complexity. The Verilog Language section contains exercises for learning the basic Verilog language features, while the "Circuit" section has exercises for using Verilog to create circuits; as problems from these two categories should be done concurrently (practicing new language features while circuits you create become more complex).

Topics

Getting Started
Using HDLBits
Verilog Language

vivado仿真



■ Vivado仿真功能概述

- 仿真在FPGA开发中常用的功能，通过给设计注入激励和观察输出结果，验证设计的功能性。
- 按设计阶段仿真分三个阶段进行：
 1. **RTL级行为仿真**: 在综合和实现前便可验证设计，用来检查代码语法和验证代码是否像设计者想要的功能一样工作，早期的行为级仿真可以尽早发现问题；
 2. **综合后仿真**: 使用综合网表仿真，验证综合后设计满足功能需求。该阶段仿真不太常用，可以用时序仿真来估计时间；功能仿真由层次化的网表组成，最底层由Xilinx原语构成；
 3. **实现后仿真**: 可以进行**功能仿真**和**时序仿真**，且与FPGA硬件上的工作情况最为接近，确保实现后设计满足功能和时序要求。



■ 功能仿真与时序仿真

- **功能仿真**仅对逻辑功能进行测试模拟，仿真过程不涉及具体器件的硬件性能；**时序仿真**包含器件硬件的特性参数，精度更高
- 时序仿真相比功能仿真要耗费大量的时间，但是可以检测到功能仿真无法检测的问题，比如因不同仿真器对语法的不同解释造成功能发生改变、错误的时序约束等。

vivado仿真



■ 使用Testbench激励文件

- TestBench也是由HDL语言代码编写，它实例化了需要仿真的设计，生成设计所需要的激励信号，监测设计输出结果并检查功能的正确性；
- 简单的TestBench可以仅仅将激励顺序地加载到设计的输入信号上；
- 复杂的TestBench可能会包含子程序调用、从外部文件读取激励信号、条件化激励和其它更多复杂的结构。

■ Testbench编写注意事项

- 在Verilog TestBench中总是使用timescale规定时间，如`timescale 1ns/1ps
- 注意Testbench编写的代码大部分不可综合，不能与可综合语法混淆

```
1 `timescale 1ns / 1ps // 单位时间/时间精度
2 module uart_tb();
3
4     //定义信号
5     reg Clk;
6     reg Rst_n;
7     wire uart_tx;
8
9     //实例化
10    uart_tx uart_tx_tb(
11        .clk(Clk),           // 100MHz 系统时钟
12        .reset(Rst_n),      // 复位信号
13        .tx(uart_tx)        // 输出端口
14    );
15
16    initial Clk = 1;       //初始时钟
17    always #5 Clk = ~Clk; //每隔5ns反转一次，即产生100MHz时钟
18    initial begin
19        Rst_n = 0;
20        #200;
21        Rst_n = 1;
22    end
23 endmodule
```

vivado仿真



Untitled 2



```
35 //生成仿真测试波形
36 initial begin
37     //初始化需要测试的信号
38     key_all = 0;
39     key_a = 0;
40     key_b = 0;
41     //1000ns后key_all置为高电平
42     #1000
43     key_all = 1;
44     #1000
45     key_a = 1;
46     key_b = 0;
47     #1000
48     key_a = 0;
49     key_b = 1;
50     #1000
51     key_a = 1;
52     key_b = 1;
53     #1000
54     key_a = 0;
55     key_b = 0;
56     #1000
57     key_all = 0;
58
59 end
```



THANKS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第三讲 (3.3 Lab1 LED控制实验)

2025年3月18日

Lab1LED控制实验



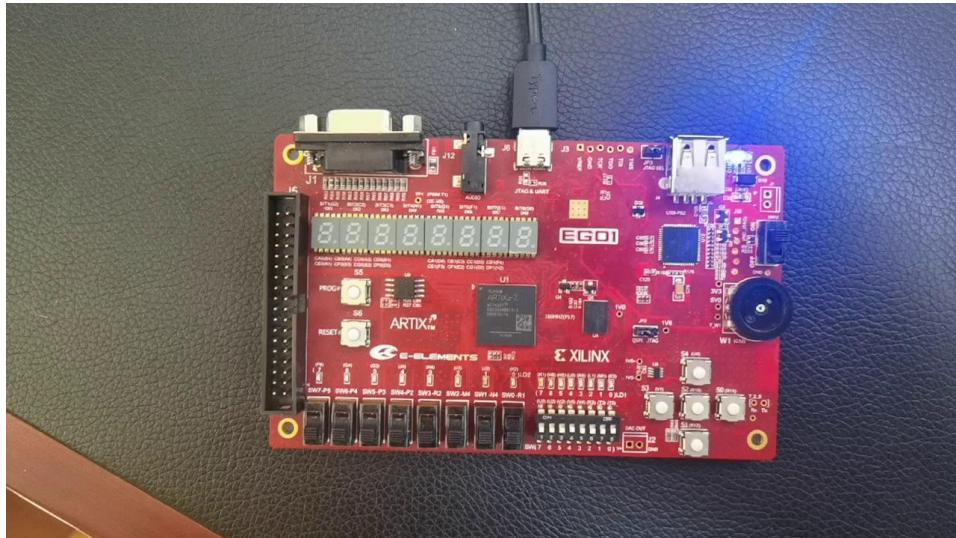
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■实验内容

- 设计并实现一套简易双控灯
- 三个按键：总控开关、灯控开关A、灯控开关B
- 一个LED
- 总控开关实现LED的整体控制
- 灯控开关A&B实现LED的双控

■实验目的

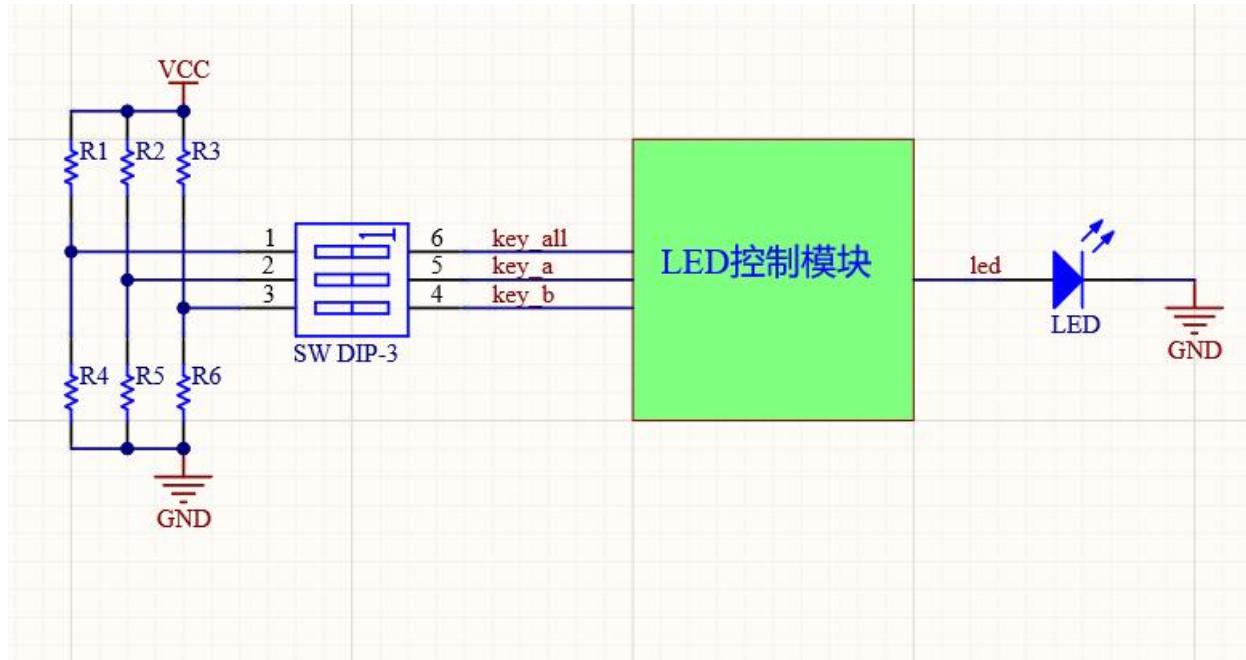
- 进一步熟悉Vivado设计流程
- 熟悉Verilog程序框架
- 学会Verilog的 always,if else语句
- 学会使用always@(*)语法实现组合逻辑电路
- 了解Testbench的使用



Lab1 LED控制实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



LED控制实验-电路图

Lab1LED控制实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

序号	信号名称	输入/输出	信号描述	FPGA引脚	说明
1	key_all	输入	总控开关	R1	总控开关输入信号，高电平有效
2	key_a	输入	灯控开关A	N4	灯控开关A信号，高电平有效
3	key_b	输入	灯控开关B	M4	灯控开关B信号，高电平有效
4	led	输出	LED灯	K2	LED灯，高电平点亮

led控制实验-输入输出端口表

Lab1LED控制实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

key_all	key_a	key_b	led
0	X	X	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

led控制实验-真值表



LED控制实验 实验步骤

Lab1控制实验



■ 易错点汇总

- 设计文件与约束文件中信号定义的一致性
- 设计文件修改后记得保存
- VS Code设置问题
- 注意不要使用中文符号
- 设计文件保存后才会进行语法检查
- 学会使用分析工具：RTL原理图，综合后原理图，器件实现

```
File Edit Selection View Go Run Terminal Help
Lab1_led.v
C: > Xilinx > MyProject_2022 > Lab1_led > Lab1_led.srcs > sources_1 > new > Lab1_led.v
1 `timescale 1ns / 1ps
2 ///////////////////////////////////////////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 2022/03/07 11:22:54
7 // Design Name:
8 // Module Name: Lab1_led
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22 module Lab1_led(
23     input key_all,
24     input key_a,
25     input key_b,
26     output led
27 );
28
29 endmodule
30
31
```

Lab1控制实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

The screenshot shows the Vivado IDE interface. On the left, the 'Cell Properties' window is open for a cell named 'led_i'. In the center, two synthesis reports are displayed side-by-side: 'Vivado Synthesis Defaults' and 'Vivado Implementation Defaults'. Both reports show 'None' for their respective incremental synthesis and implementation strategies. Below these reports are two tabs: 'DRC Violations' and 'Timing'. At the bottom, the 'Messages' tab is active, showing a warning message: '[Constraints 18-5210] No constraints selected for write.' This message includes a resolution note about constraints being ignored during synthesis.

1. 重视警告信息
2. 上述警告在Vivado 2018.2-2019.2版本会出现，没有实际影响
3. 2020.1版本之后不会出现

Lab1控制实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 实验步骤

- 创建工程项目，工程名：Lab1_led
- 完成设计源代码输入，文件名：Lab1_led.v
- 完成设计约束文件输入，文件名：Lab1_led.xdc
- 完成仿真文件输入，文件名：Lab1_led_sim.v
- 综合 (synthesis)
- 仿真 (simulation)
- 实现 (implemetation)
- 生成bit文件
- 下载程序，查看程序运行效果

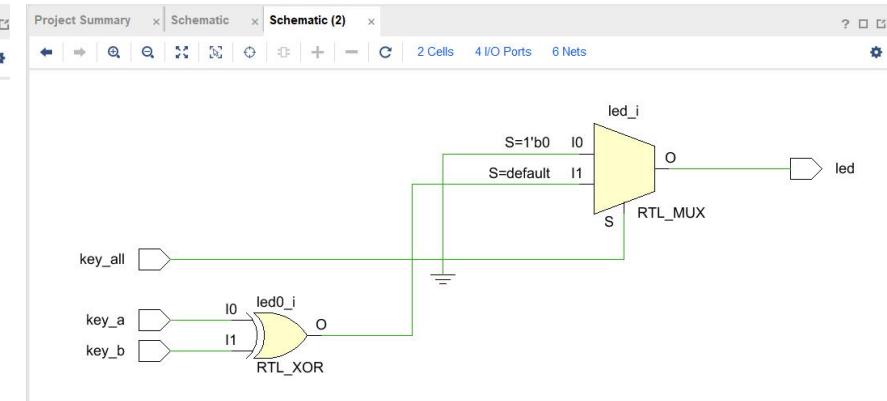
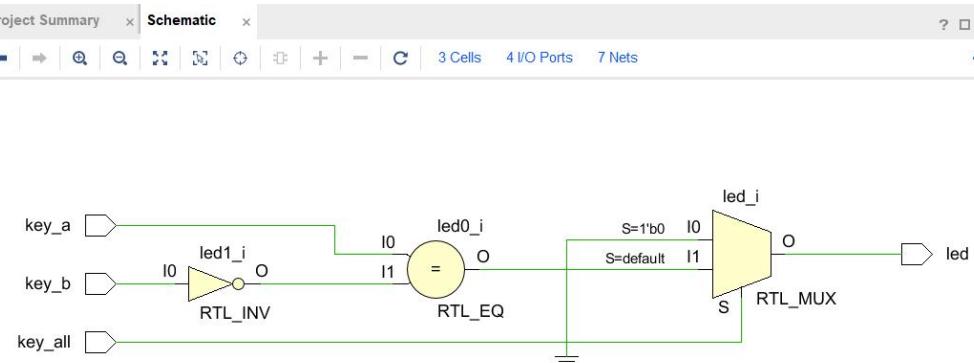
Lab1控制实验



■ 输入文件 (Lab1_led.v)

- 要求使用`always@(*)`语法规实现
- `always`块中的使用的输出信号需要定义成`reg`类型: **output reg led**
- 可以使用`if else`语法实现

```
module Lab1_led(  
    input key_all,  
    input key_a,  
    input key_b,  
    output reg led  
);
```



两种不同的RTL电路形式

Lab1控制实验



■ 约束文件 (Lab1_led.xdc)

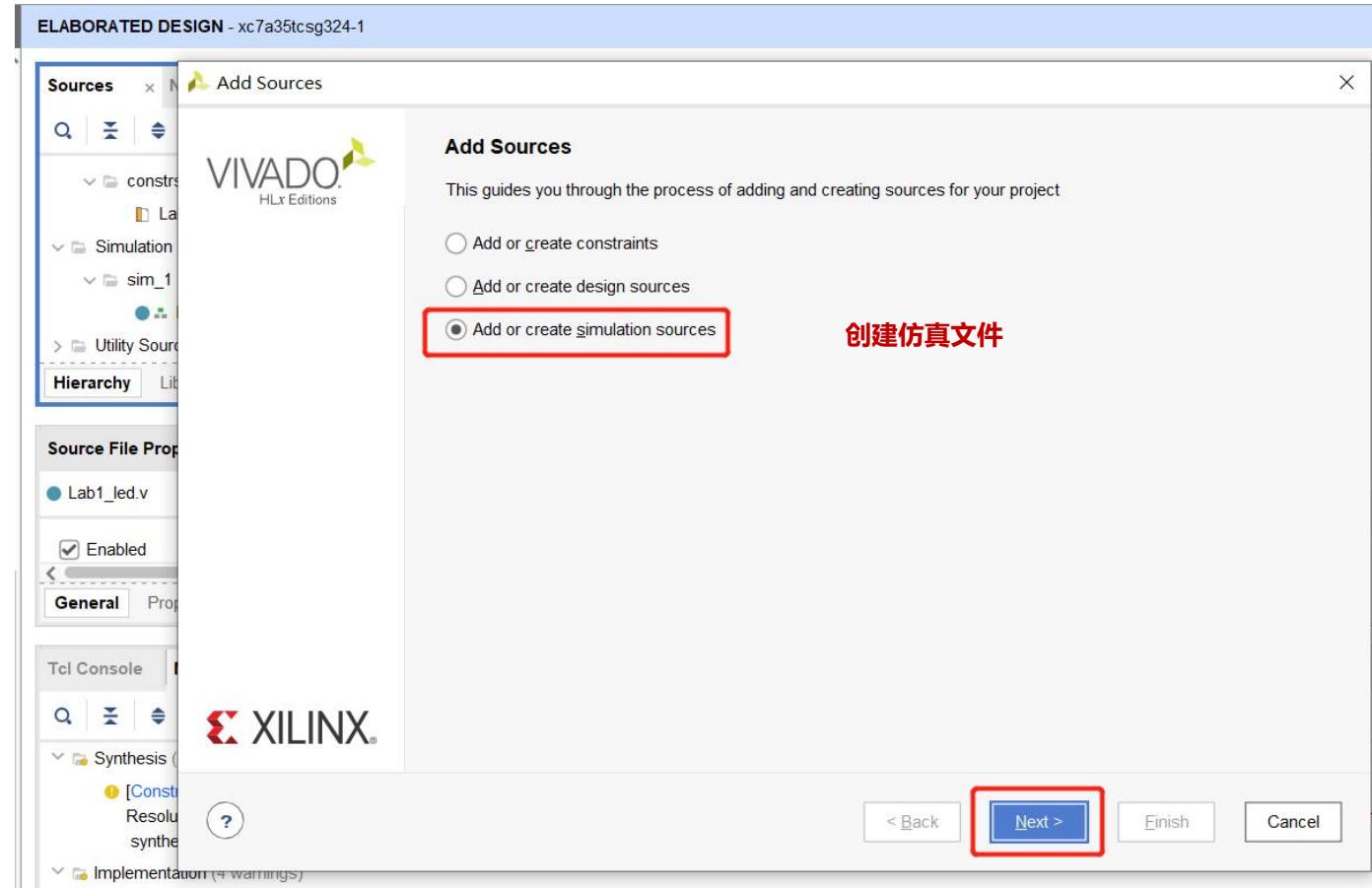
- key_all配置到R1引脚
- key_a配置到N4引脚
- key_b配置到M4引脚
- led配置到K2引脚

≡ Lab1_led.xdc ×

C: > Xilinx > MyProject_2022 > Lab1_led > Lab1_led.srcs > constrs_1 > new > ≡ Lab1_led.xdc

```
1 set_property -dict {PACKAGE_PIN M4 IO_STANDARD LVCMOS33} [get_ports {key_b}]
2 set_property -dict {PACKAGE_PIN N4 IO_STANDARD LVCMOS33} [get_ports {key_a}]
3 set_property -dict {PACKAGE_PIN R1 IO_STANDARD LVCMOS33} [get_ports {key_all}]
4
5 set_property -dict {PACKAGE_PIN K2 IO_STANDARD LVCMOS33} [get_ports {led}]
```

Lab1控制实验



Lab1控制实验



ELABORATED DESIGN - xc7a35tcsg324-1

Add Sources

Add or Create Simulation Sources

Specify simulation specific HDL files, or directories containing HDL files, to add to your project. Create a new source file on disk and add it to your project.

Specify simulation set: sim_1

创建仿真文件

Create Source File

Create a new source file and add it to your project.

File type: Verilog

File name: Lab1_led_sim

File location: <Local to Project>

OK Cancel

Add Files Add Directories Create File

Scan and add RTL include files into project

Copy sources into project

Add sources from subdirectories

Include all design sources for simulation

< Back Next > Finish Cancel

This later case is used when

Lab1控制实验



ELABORATED DESIGN - xc7a35tcsg324-1

Sources x Netlist ? □ Updating 0 Project Summary

Sources x Netlist ? □ Updating 0 Project Summary

Define Module

Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is checked.
Ports with blank names will not be written.

Module Definition

Module name: Lab1_led_sim

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
	input	<input type="checkbox"/>	0	0

仿真文件不需要输入输出端口

OK Cancel

Source File Properties

Lab1_led.v

Enabled

General Properties

Tcl Console Messages Log

Synthesis (1 warning)

Lab1控制实验



國科大杭州高華研究院 Hangzhou Institute for Advanced Study, UCAS

ELABORATED DESIGN - xc7a35tcsg324-1

The screenshot shows the Xilinx Vivado IDE interface. The top navigation bar displays "ELABORATED DESIGN - xc7a35tcsg324-1".

Sources (tab selected):

- Netlist
- Lab1_led.xdc
- Simulation Sources (2)
 - sim_1 (2)
 - Lab1_led (Lab1_led.v)
 - Lab1_led_sim (Lab1_led_sim.v) (highlighted with a red box)
- Utility Sources

Hierarchy tab is selected.

Source File Properties (tab selected):

- Lab1_led_sim.v
- Enabled (checkbox checked)

General tab is selected.

Project Summary (tab selected):

Overview | Dashboard

Settings tab is selected.

Setting	Value
Project name:	Lab1_led
Project location:	C:/Xilinx/MyProject_2022/Lab1_led
Product family:	Artix-7
Project part:	xc7a35tcsg324-1
Top module name:	Lab1_led
Target language:	Verilog
Simulator language:	Mixed

Synthesis (tab selected):

Status	Message
Status:	Complete (green checkmark)
Messages:	1 warning (yellow exclamation mark)

Lab1控制实验



■ 仿真文件 (Lab1_led_sim.v)

- 定义仿真输入输出信号，输入信号使用reg类型，输出信号使用wire类型
- 例化仿真测试模块
- 生成仿真测试波形

子模块

```
module lab1_led(  
    input      key_all,  
    input      key_a,  
    input      key_b,  
    output reg led  
) ;
```

子模块名

```
Lab1_led inst1(  
    .key_all(key_all),  
    .key_a(key_a),  
    .key_b(key_b),  
    .led(led))
```

子模块端口信号

例化模块端口信号

```
22  
23 module Lab1_led_sim(  
24  
25 );  
26 // 定义仿真输入、输出信号  
27 // 输入信号使用reg类型，输出信号使用wire类型  
28 reg key_all;  
29 reg key_a;  
30 reg key_b;  
31 wire led;  
32  
33 //例化仿真测试模块  
34 Lab1_led inst1(.key_all(key_all),.key_a(key_a),.key_b(key_b),.led(led));  
35  
36 //生成仿真测试波形  
37 initial begin  
38     //初始化需要测试的信号  
39     key_all = 0;  
40     key_a = 0;  
41     key_b = 0;  
42     //1000ns后key_all置为高电平  
43     #1000  
44     key_all = 1;  
45     #1000  
46     key_a = 1;  
47     key_b = 0;  
48     #1000  
49     key_a = 0;  
50     key_b = 1;  
51     #1000  
52     key_a = 1;  
53     key_b = 1;  
54     #1000  
55     key_a = 0;  
56     key_b = 0;  
57     #1000  
58     key_all = 0;  
59  
60 end  
61  
62 endmodule
```

Lab1控制实验



Lab1_led - [C:/Xilinx/MyProject_2022/Lab1_led/Lab1_led.xpr] - Vivado 2019.1

File Edit Flow Tools Reports Window Layout View Help Quick Access

Flow Navigator

- Create Block Design
- Open Block Design
- Generate Block Design
- SIMULATION
 - Run Simulation
 - Run Behavioral Simulation **运行行为仿真**
 - Run Post-Synthesis Functional Simulation
 - Run Post-Synthesis Timing Simulation
 - Run Post-Implementation Functional Simulation
 - Run Post-Implementation Timing Simulation
- RTL ANALYSIS
 - Open Elab
 - Report
 - Report
 - Report Noise
 - Schematic
- SYNTHESIS
 - Run Synthesis
 - Open Synthesized Design
 - Constraints Wizard

ELABORATED DESIGN - xc7a35tcsg324-1

Sources Netlist

- Lab1_led (Lab1_led.v)
- Constraints (1)
- Simulation Sources (1)

Project Summary

Overview | Dashboard

Settings Edit

Project name: Lab1_led
Project location: C:/Xilinx/MyProject_2022/Lab1_led
Product family: Artix-7
Project part: xc7a35tcsg324-1
Top module name: Lab1_led
Target language: Verilog
Simulator language: Mixed

Synthesis

Status: Complete
Messages: 1 warning

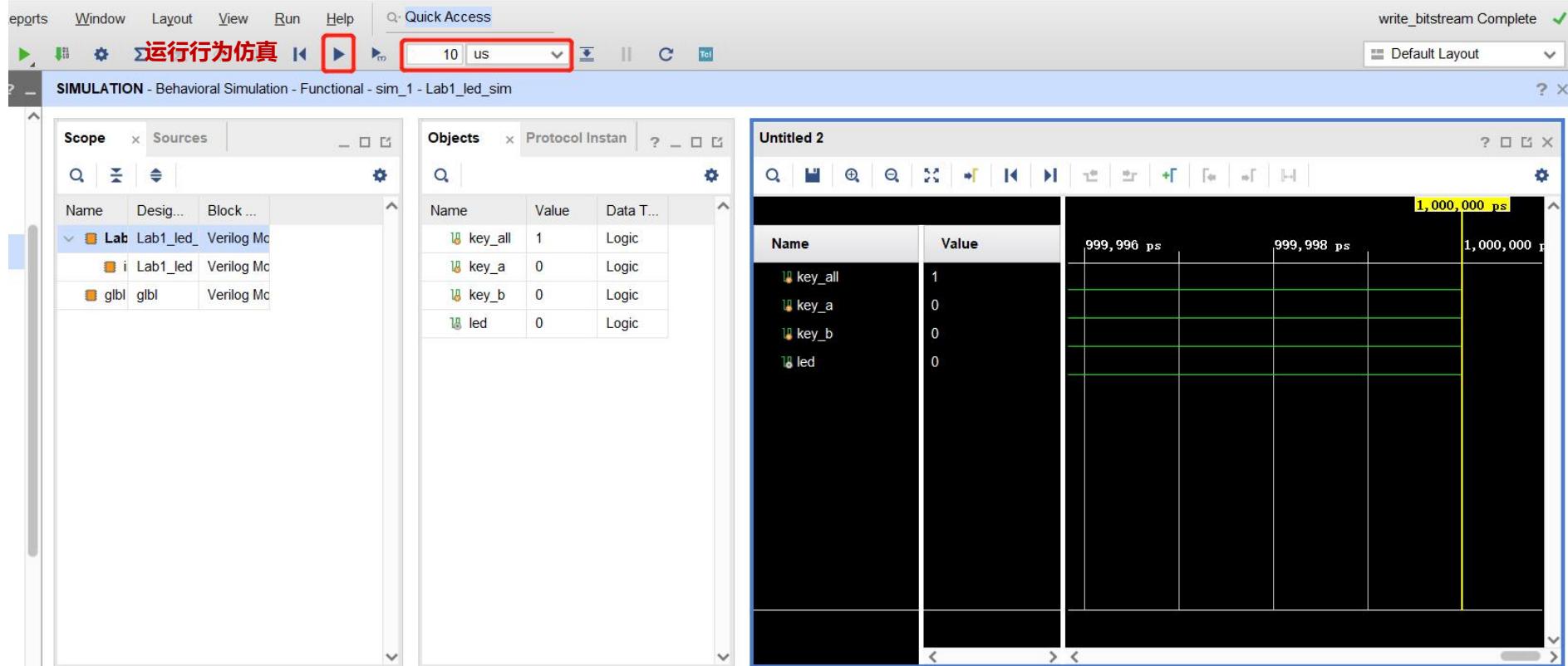
Tcl Console Messages Log Reports Design Runs

Warning (5) Info (144) Status (255) Show All

Lab1控制实验



2022/Lab1_led/Lab1_led.xpr] - Vivado 2019.1



Lab1控制实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS



核对仿真结果



- 综合 (synthesis)
- 实现 (implemetation)
- 生成bit文件
- 下载程序，查看程序运行效果



**不使用always@(*)语法，使用assign语法实现
如何仅使用一条语句完成?
查看两种方式下电路的区别**

课后作业：HDLbits题目



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

▶ Getting Started

▼ Verilog Language

▶ Basics

▶ Vectors

▶ Modules; Hierarchy

▼ Procedures

○ Always blocks
(combinational)

Always blocks (clocked)

○ If statement

○ If statement latches

○ Case statement

○ Priority encoder

○ Priority encoder with casez

○ Avoiding latches

▶ More Verilog Features

▶ Circuits

▶ Verification: Reading Simulations

▶ Verification: Writing Testbenches

▶ CS450

▶ Getting Started

▶ Verilog Language

▼ Circuits

▶ Combinational Logic

▼ Sequential Logic

▶ Latches and Flip-Flops

▼ Counters

Four-bit binary counter

○ Decade counter

○ Decade counter again

○ Slow decade counter

○ Counter 1-12

○ Counter 1000

○ 4-digit decimal counter

○ 12-hour clock

▶ Shift Registers

▶ More Circuits

▶ Finite State Machines

▶ Building Larger Circuits

▶ Verification: Reading Simulations

▶ Verification: Writing Testbenches

▶ CS450

▶ Getting Started

▶ Verilog Language

▼ Circuits

▶ Combinational Logic

▼ Sequential Logic

▶ Latches and Flip-Flops

▶ Counters

▼ Shift Registers

4-bit shift register

○ Left/right rotator

○ Left/right arithmetic shift
by 1 or 8

○ 5-bit LFSR

○ 3-bit LFSR

○ 32-bit LFSR

○ Shift register

○ Shift register

○ 3-input LUT

▶ More Circuits

▶ Finite State Machines

▶ Building Larger Circuits

▶ Verification: Reading Simulations

▶ Verification: Writing Testbenches

▶ CS450

下节预告



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Lab2实验内容

Lab2分频计数器

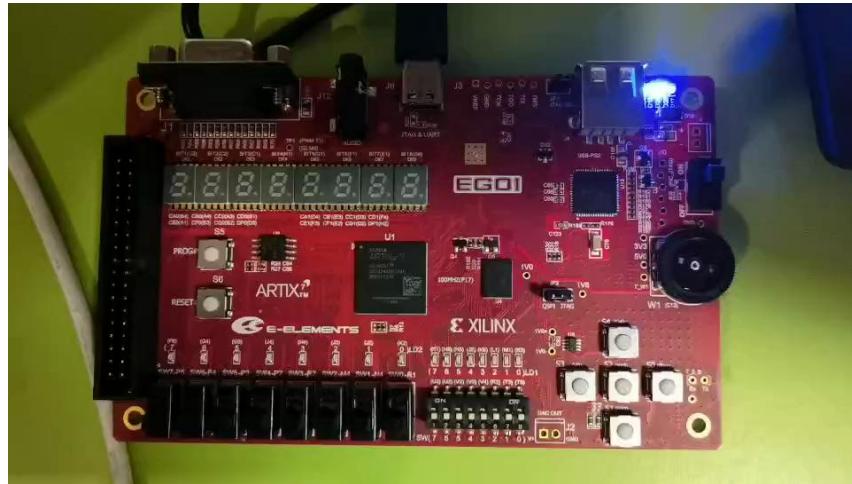


■实验内容

- 选择一颗LED，按照1Hz的频率进行闪烁
- 输入时钟为系统时钟 (100MHz)

■实验目的

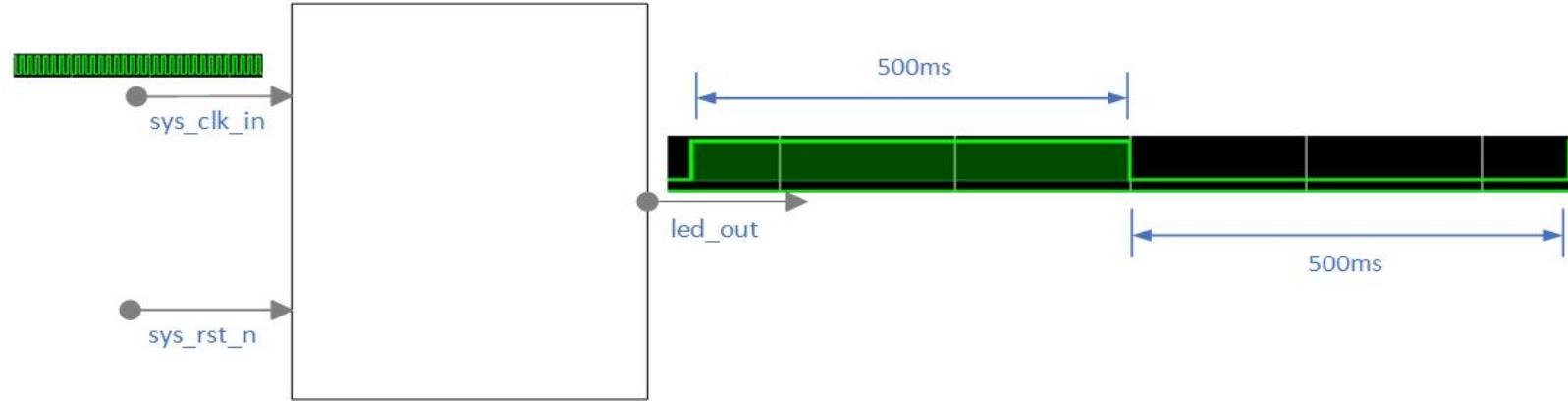
- 进一步熟悉Vivado设计工具
- 熟悉Verilog程序框架
- 学会常规计数器的设计
- 熟悉Testbench的使用



Lab2分频计数器



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



■ 电路结构

- 使用累加器对输入脉冲进行计数
- 复位信号有效时计数器清零
- 100MHz信号的时钟周期为10ns
- 计数器每500ms清零一次，同时将`led_out`翻转一次

$$\text{计数值} = \frac{500\text{ms}}{10\text{ns}} = 50 \times 10^6$$

Lab2分频计数器



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

序号	信号名称	输入/输出	信号描述	FPGA引脚	说明
1	sys_clk_in	输入	系统时钟输入	P17	100MHz输入
2	sys_rst_n	输入	系统复位输入	P15	低电平复位
3	led_out	输出	led控制信号输出	K3	高电平点亮

输入输出端口表



THANKS



國科大杭州高等研究院
Hangzhou Institute for Advanced Study, UCAS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第四讲 (4.1 Verilog语法1)

2025年3月25日



1. Verilog代码风格
2. Verilog语法知识
3. 层次化程序结构
4. IP核使用



程序设计是一门艺术吗？

Verilog代码风格



■ 艺术

- 艺术是广泛的人类活动或其产品
- 涉及**创造性的想象力**, 旨在表达**技术熟练程度、美感、情感力量或概念观念**

■ 程序设计

- 程序设计像艺术一样, **深不可测、奥妙无穷**
- 程序员像艺术家一样, **有发挥创造性的无限空间**
- 一个好的算法应该像一段音乐, 一个好的程序应该像一部文学作品

■ Verilog语言之美

- 通过代码风格体现
- 整洁高效, 优雅有序

■ 美是一种修养

- 严谨精确, 熟练掌握语言的规则
- 刻苦实践, 熟能生巧, 习以为常, 知行合一



```
35 // 直接将多帧的规则
36 always @(posedge clk or posedge rst) begin
37   if (rst==1'b1) begin
38     state<-IDLE;
39   end
40   else begin
41     case(state)
42       43
43       44
44       45
45       46
46       47
47       48
48       49
49       50
50       51
51       52
52       53
53       54
54       55
55       56
56       57
57       58
58       59
59       60
60       61
61       62
62       63
63       64
64       65
65       66
66       67
67       68
68       69
69       70
70       71
71       72
72       73
73       74
74       75
75       76
76       77
77       78
78       79
79       80
80       81
81       82
82       83
83       84
84       85
85       86
86       87
87       88
88       89
89       90
90       91
91       92
92       93
93       94
94       95
95       96
96       97
97       98
98       99
99       100
100      101
101      102
102      103
103      104
104      105
105      106
106      107
107      108
108      109
109      110
110      111
111      112
112      113
113      114
114      115
115      116
116      117
117      118
118      119
119      120
120      121
121      122
122      123
123      124
124      125
125      126
126      127
127      128
128      129
129      130
130      131
131      132
132      133
133      134
134      135
135      136
136      137
137      138
138      139
139      140
140      141
141      142
142      143
143      144
144      145
145      146
146      147
147      148
148      149
149      150
150      151
151      152
152      153
153      154
154      155
155      156
156      157
157      158
158      159
159      160
160      161
161      162
162      163
163      164
164      165
165      166
166      167
167      168
168      169
169      170
170      171
171      172
172      173
173      174
174      175
175      176
176      177
177      178
178      179
179      180
180      181
181      182
182      183
183      184
184      185
185      186
186      187
187      188
188      189
189      190
190      191
191      192
192      193
193      194
194      195
195      196
196      197
197      198
198      199
199      200
200      201
201      202
202      203
203      204
204      205
205      206
206      207
207      208
208      209
209      210
210      211
211      212
212      213
213      214
214      215
215      216
216      217
217      218
218      219
219      220
220      221
221      222
222      223
223      224
224      225
225      226
226      227
227      228
228      229
229      230
230      231
231      232
232      233
233      234
234      235
235      236
236      237
237      238
238      239
239      240
240      241
241      242
242      243
243      244
244      245
245      246
246      247
247      248
248      249
249      250
250      251
251      252
252      253
253      254
254      255
255      256
256      257
257      258
258      259
259      260
260      261
261      262
262      263
263      264
264      265
265      266
266      267
267      268
268      269
269      270
270      271
271      272
272      273
273      274
274      275
275      276
276      277
277      278
278      279
279      280
280      281
281      282
282      283
283      284
284      285
285      286
286      287
287      288
288      289
289      290
290      291
291      292
292      293
293      294
294      295
295      296
296      297
297      298
298      299
299      300
300      301
301      302
302      303
303      304
304      305
305      306
306      307
307      308
308      309
309      310
310      311
311      312
312      313
313      314
314      315
315      316
316      317
317      318
318      319
319      320
320      321
321      322
322      323
323      324
324      325
325      326
326      327
327      328
328      329
329      330
330      331
331      332
332      333
333      334
334      335
335      336
336      337
337      338
338      339
339      340
340      341
341      342
342      343
343      344
344      345
345      346
346      347
347      348
348      349
349      350
350      351
351      352
352      353
353      354
354      355
355      356
356      357
357      358
358      359
359      360
360      361
361      362
362      363
363      364
364      365
365      366
366      367
367      368
368      369
369      370
370      371
371      372
372      373
373      374
374      375
375      376
376      377
377      378
378      379
379      380
380      381
381      382
382      383
383      384
384      385
385      386
386      387
387      388
388      389
389      390
390      391
391      392
392      393
393      394
394      395
395      396
396      397
397      398
398      399
399      400
400      401
401      402
402      403
403      404
404      405
405      406
406      407
407      408
408      409
409      410
410      411
411      412
412      413
413      414
414      415
415      416
416      417
417      418
418      419
419      420
420      421
421      422
422      423
423      424
424      425
425      426
426      427
427      428
428      429
429      430
430      431
431      432
432      433
433      434
434      435
435      436
436      437
437      438
438      439
439      440
440      441
441      442
442      443
443      444
444      445
445      446
446      447
447      448
448      449
449      450
450      451
451      452
452      453
453      454
454      455
455      456
456      457
457      458
458      459
459      460
460      461
461      462
462      463
463      464
464      465
465      466
466      467
467      468
468      469
469      470
470      471
471      472
472      473
473      474
474      475
475      476
476      477
477      478
478      479
479      480
480      481
481      482
482      483
483      484
484      485
485      486
486      487
487      488
488      489
489      490
490      491
491      492
492      493
493      494
494      495
495      496
496      497
497      498
498      499
499      500
500      501
501      502
502      503
503      504
504      505
505      506
506      507
507      508
508      509
509      510
510      511
511      512
512      513
513      514
514      515
515      516
516      517
517      518
518      519
519      520
520      521
521      522
522      523
523      524
524      525
525      526
526      527
527      528
528      529
529      530
530      531
531      532
532      533
533      534
534      535
535      536
536      537
537      538
538      539
539      540
540      541
541      542
542      543
543      544
544      545
545      546
546      547
547      548
548      549
549      550
550      551
551      552
552      553
553      554
554      555
555      556
556      557
557      558
558      559
559      560
560      561
561      562
562      563
563      564
564      565
565      566
566      567
567      568
568      569
569      570
570      571
571      572
572      573
573      574
574      575
575      576
576      577
577      578
578      579
579      580
580      581
581      582
582      583
583      584
584      585
585      586
586      587
587      588
588      589
589      590
590      591
591      592
592      593
593      594
594      595
595      596
596      597
597      598
598      599
599      600
600      601
601      602
602      603
603      604
604      605
605      606
606      607
607      608
608      609
609      610
610      611
611      612
612      613
613      614
614      615
615      616
616      617
617      618
618      619
619      620
620      621
621      622
622      623
623      624
624      625
625      626
626      627
627      628
628      629
629      630
630      631
631      632
632      633
633      634
634      635
635      636
636      637
637      638
638      639
639      640
640      641
641      642
642      643
643      644
644      645
645      646
646      647
647      648
648      649
649      650
650      651
651      652
652      653
653      654
654      655
655      656
656      657
657      658
658      659
659      660
660      661
661      662
662      663
663      664
664      665
665      666
666      667
667      668
668      669
669      670
670      671
671      672
672      673
673      674
674      675
675      676
676      677
677      678
678      679
679      680
680      681
681      682
682      683
683      684
684      685
685      686
686      687
687      688
688      689
689      690
690      691
691      692
692      693
693      694
694      695
695      696
696      697
697      698
698      699
699      700
700      701
701      702
702      703
703      704
704      705
705      706
706      707
707      708
708      709
709      710
710      711
711      712
712      713
713      714
714      715
715      716
716      717
717      718
718      719
719      720
720      721
721      722
722      723
723      724
724      725
725      726
726      727
727      728
728      729
729      730
730      731
731      732
732      733
733      734
734      735
735      736
736      737
737      738
738      739
739      740
740      741
741      742
742      743
743      744
744      745
745      746
746      747
747      748
748      749
749      750
750      751
751      752
752      753
753      754
754      755
755      756
756      757
757      758
758      759
759      760
760      761
761      762
762      763
763      764
764      765
765      766
766      767
767      768
768      769
769      770
770      771
771      772
772      773
773      774
774      775
775      776
776      777
777      778
778      779
779      780
780      781
781      782
782      783
783      784
784      785
785      786
786      787
787      788
788      789
789      790
790      791
791      792
792      793
793      794
794      795
795      796
796      797
797      798
798      799
799      800
800      801
801      802
802      803
803      804
804      805
805      806
806      807
807      808
808      809
809      810
810      811
811      812
812      813
813      814
814      815
815      816
816      817
817      818
818      819
819      820
820      821
821      822
822      823
823      824
824      825
825      826
826      827
827      828
828      829
829      830
830      831
831      832
832      833
833      834
834      835
835      836
836      837
837      838
838      839
839      840
840      841
841      842
842      843
843      844
844      845
845      846
846      847
847      848
848      849
849      850
850      851
851      852
852      853
853      854
854      855
855      856
856      857
857      858
858      859
859      860
860      861
861      862
862      863
863      864
864      865
865      866
866      867
867      868
868      869
869      870
870      871
871      872
872      873
873      874
874      875
875      876
876      877
877      878
878      879
879      880
880      881
881      882
882      883
883      884
884      885
885      886
886      887
887      888
888      889
889      890
890      891
891      892
892      893
893      894
894      895
895      896
896      897
897      898
898      899
899      900
900      901
901      902
902      903
903      904
904      905
905      906
906      907
907      908
908      909
909      910
910      911
911      912
912      913
913      914
914      915
915      916
916      917
917      918
918      919
919      920
920      921
921      922
922      923
923      924
924      925
925      926
926      927
927      928
928      929
929      930
930      931
931      932
932      933
933      934
934      935
935      936
936      937
937      938
938      939
939      940
940      941
941      942
942      943
943      944
944      945
945      946
946      947
947      948
948      949
949      950
950      951
951      952
952      953
953      954
954      955
955      956
956      957
957      958
958      959
959      960
960      961
961      962
962      963
963      964
964      965
965      966
966      967
967      968
968      969
969      970
970      971
971      972
972      973
973      974
974      975
975      976
976      977
977      978
978      979
979      980
980      981
981      982
982      983
983      984
984      985
985      986
986      987
987      988
988      989
989      990
990      991
991      992
992      993
993      994
994      995
995      996
996      997
997      998
998      999
999      1000
1000      1001
1001      1002
1002      1003
1003      1004
1004      1005
1005      1006
1006      1007
1007      1008
1008      1009
1009      1010
1010      1011
1011      1012
1012      1013
1013      1014
1014      1015
1015      1016
1016      1017
1017      1018
1018      1019
1019      1020
1020      1021
1021      1022
1022      1023
1023      1024
1024      1025
1025      1026
1026      1027
1027      1028
1028      1029
1029      1030
1030      1031
1031      1032
1032      1033
1033      1034
1034      1035
1035      1036
1036      1037
1037      1038
1038      1039
1039      1040
1040      1041
1041      1042
1042      1043
1043      1044
1044      1045
1045      1046
1046      1047
1047      1048
1048      1049
1049      1050
1050      1051
1051      1052
1052      1053
1053      1054
1054      1055
1055      1056
1056      1057
1057      1058
1058      1059
1059      1060
1060      1061
1061      1062
1062      1063
1063      1064
1064      1065
1065      1066
1066      1067
1067      1068
1068      1069
1069      1070
1070      1071
1071      1072
1072      1073
1073      1074
1074      1075
1075      1076
1076      1077
1077      1078
1078      1079
1079      1080
1080      1081
1081      1082
1082      1083
1083      1084
1084      1085
1085      1086
1086      1087
1087      1088
1088      1089
1089      1090
1090      1091
1091      1092
1092      1093
1093      1094
1094      1095
1095      1096
1096      1097
1097      1098
1098      1099
1099      1100
1100      1101
1101      1102
1102      1103
1103      1104
1104      1105
1105      1106
1106      1107
1107      1108
1108      1109
1109      1110
1110      1111
1111      1112
1112      1113
1113      1114
1114      1115
1115      1116
1116      1117
1117      1118
1118      1119
1119      1120
1120      1121
1121      1122
1122      1123
1123      1124
1124      1125
1125      1126
1126      1127
1127      1128
1128      1129
1129      1130
1130      1131
1131      1132
1132      1133
1133      1134
1134      1135
1135      1136
1136      1137
1137      1138
1138      1139
1139      1140
1140      1141
1141      1142
1142      1143
1143      1144
1144      1145
1145      1146
1146      1147
1147      1148
1148      1149
1149      1150
1150      1151
1151      1152
1152      1153
1153      1154
1154      1155
1155      1156
1156      1157
1157      1158
1158      1159
1159      1160
1160      1161
1161      1162
1162      1163
1163      1164
1164      1165
1165      1166
1166      1167
1167      1168
1168      1169
1169      1170
1170      1171
1171      1172
1172      1173
1173      1174
1174      1175
1175      1176
1176      1177
1177      1178
1178      1179
1179      1180
1180      1181
1181      1182
1182      1183
1183      1184
1184      1185
1185      1186
1186      1187
1187      1188
1188      1189
1189      1190
1190      1191
1191      1192
1192      1193
1193      1194
1194      1195
1195      1196
1196      1197
1197      1198
1198      1199
1199      1200
1200      1201
1201      1202
1202      1203
1203      1204
1204      1205
1205      1206
1206      1207
1207      1208
1208      1209
1209      1210
1210      1211
1211      1212
1212      1213
1213      1214
1214      1215
1215      1216
1216      1217
1217      1218
1218      1219
1219      1220
1220      1221
1221      1222
1222      1223
1223      1224
1224      1225
1225      1226
1226      1227
1227      1228
1228      1229
1229      1230
1230      1231
1231      1232
1232      1233
1233      1234
1234      1235
1235      1236
1236      1237
1237      1238
1238      1239
1239      1240
1240      1241
1241      1242
1242      1243
1243      1244
1244      1245
1245      1246
1246      1247
1247      1248
1248      1249
1249      1250
1250      1251
1251      1252
1252      1253
1253      1254
1254      1255
1255      1256
1256      1257
1257      1258
1258      1259
1259      1260
1260      1261
1261      1262
1262      1263
1263      1264
1264      1265
1265      1266
1266      1267
1267      1268
1268      1269
1269      1270
1270      1271
1271      1272
1272      1273
1273      1274
1274      1275
1275      1276
1276      1277
1277      1278
1278      1279
1279      1280
1280      1281
1281      1282
1282      1283
12
```

Verilog代码风格



■ 什么是代码风格

- 代码符合一定的**格式**
- 满足**功能和性能要求为前提**, 优秀的代码风格是对设计的**精益求精**
- 规范代码和电路, 增加代码的**整洁性、可读性、可修改性、可维护性、可重用性**
- 团队合作开发的要求
- 没有固定的要求, 可以灵活运用

■ 优秀代码风格的好处

- 通常风格优秀的代码, 质量上也较为可靠
- **整洁**的代码, 意图清晰, 干净利落
- **高效**的代码, 降低出错概率
- 源代码被读懂不是因为注释, 而应该是由代码本身的**优雅**
- **有序**的代码, 验证充分, 利于重构

```
always@(posedge clk or posedge rst or posedge set) begin
    if(rst) begin q <= 0;end
    else if(set) begin q <= 1;end
    else begin q <= d;end
end
```

```
always@(posedge clk or posedge rst or posedge set) begin
    if(rst) begin
        q <= 0;
    end
    else if(set) begin
        q <= 1;
    end
    else begin
        q <= d;
    end
end
```



■ 逻辑清晰

- 采用**层次化设计**, 电路框图、信号流图、模块功能、模块接口明确
- 模块使用**高内聚、低耦合原则**, 模块功能明确、接口清晰, **降低模块间的耦合性**
- 代码**结构合理**, 参数化、通用化、IP化

■ 书写规范

- 遵循通用的书写规范要求

■ 版本控制

- 使用**CVS、VSS、SVN等版本控制工具**

■ 文档齐套

- 规范文件目录
- **设计说明文件**: 需求分析报告、总体方案报告、详细设计报告、测试报告、技术总结报告等
- **设计代码**: 设计源文件、约束文件、测试仿真文件等

Verilog代码风格



■ 通用书写规范要求

- 单条语句独立成行
- 注意代码的对齐，建议使用Tab键进行缩进对齐，与C语言类似
- 需要使用begin的语句（always、if、else if、else等），begin与该语句同行，begin后需要换行
- end要单独占据一行
- 在同一个逻辑块内不加空行，不同逻辑块间使用空行
- 表达式中间可以适当增加空格，增加代码整洁性
- 逻辑块和重要的代码行的上面添加注释
- 多体会vscode、sublime等软件的**自动代码风格**

■ 名字定义

- 使用有意义且有效的名字，尽量使用单词组合，write_en或者WriteEn，建议使用下划线方式
- 信号名不要太长，建议使用单词缩写：reset->rst，clock->clk，ready->rdy，data->dat
- 遵循通用规范：clk->时钟，rst->复位，rst_n->低电平复位

```
//-----产生输出时钟信号-----//
reg [31:0] cnt; //计数,100MHz时钟
parameter CNT_MAX = 32'd10_000_000; //时间s=cnt_MAX/100M

always @ (posedge clk or posedge rst) begin
    if (!rst) begin
        cnt <= 32'd0;
        write_en <= 1'b0;
    end
    //每当计数到最大向FIFO中写一个数
    else if (cnt == CNT_MAX) begin
        write_en <= 1'b1;
        cnt <= 32'd0;
    end
    else begin
        cnt <= cnt+1'b1;
        write_en <= 1'b0;
    end
end
```



数值表示

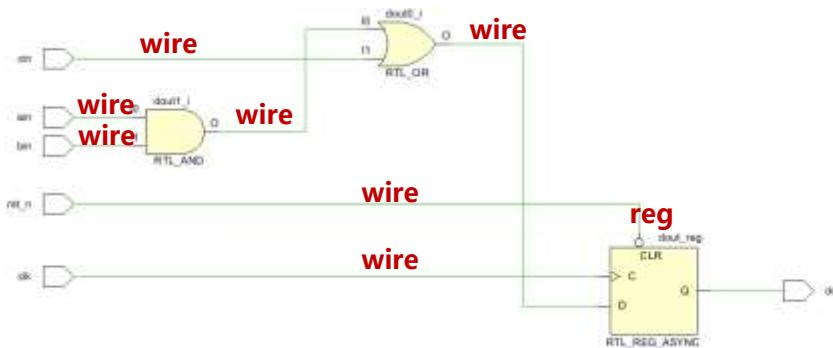
- ◆ **四种电平类型：** 0->逻辑0或假； 1->逻辑1或真； x->未知， 0或者1； z->高阻
- ◆ **整数数值类型：** 合法的**基数格式**有 4 中， 包括： **十进制**('d 或 'D), **十六进制**('h 或 'H), **二进制** ('b 或 'B) , **八进制** ('o 或 'O)
- ◆ 整数数值可以**指明位宽**: 4'b0011 //4bit数值; 32'h1234_5678 //32bit十六进制数值; 8'b0000_0000,下划线增加代码可读性
- ◆ 整数数值直接写数字**默认为十进制**， 若不指明位宽， 编译器会自动分配位宽， 建议明确位宽避免产生隐性错误
- ◆ **不同位宽信号**可以直接赋值， 有数据截断风险， 需要特别注意

Verilog 语法知识

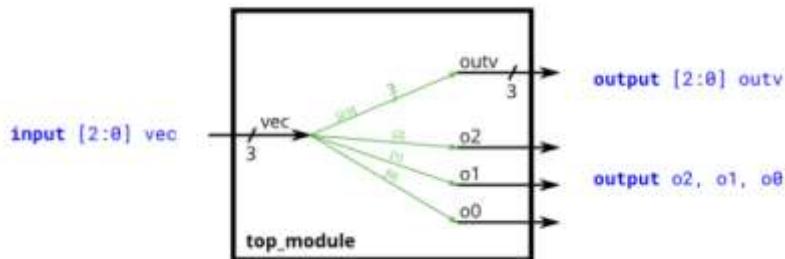


國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

数据类型



- ◆ **wire**: 硬件单元之间的物理连线，由其连接的器件输出端连续驱动
- ◆ **reg** : 表示存储单元，**保持数据原有的值**，直到被改写，reg可能被综合成边沿触发器，在组合逻辑中可能被综合成 wire 型变量
- ◆ 向量：当位宽大于 1 时，wire 或 reg 即可声明为**向量的形式**，向量可以进行**索引**，也可以进行**链接**，注意向量的方向一致



```
wire [7:0] led_out;
reg [31:0] counter;

led_out[7:6] = counter[31:30];

led_out = {led_out[0],led_out[7:1]};
```

Verilog 语法知识



■ 连续赋值语句 (always块外) (assign)

- 赋值目标只能是wire类型，电路表现形式为直接连线

■ 过程赋值 (always块内) 语句 (阻塞赋值, =)

- 赋值目标只能是reg类型，在always语句内赋值，可以对多位宽寄存器中的某一位或某几位进行赋值
- 组合逻辑中使用阻塞赋值，阻塞赋值相当于直接连线
- 如果有多条阻塞赋值语句，如果前面的赋值语句没有完成，则后面的语句就不能被执行，仿佛被阻塞了一样

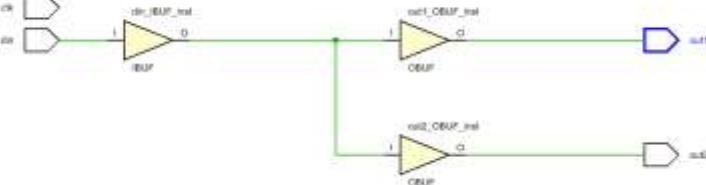
■ 过程赋值 (always块内) 语句 (非阻塞赋值, <=)

- 赋值目标只能是reg类型，在always语句内赋值，可以对多位宽寄存器中的某一位或某几位进行赋值
- 时序逻辑中使用非阻塞赋值，非阻塞赋值相当于经过寄存器
- 如果有多条非阻塞赋值语句，后面语句的执行不会受到前面语句的阻塞

```
module blocking_nonblocking(
    input din,
    input clk,
    output reg out1,
    output reg out2
);

    reg reg_din;

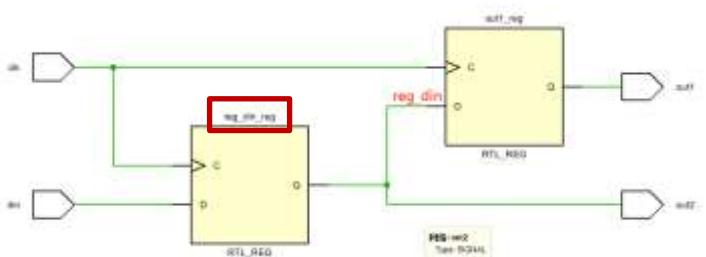
    always@(*) begin
        reg_din = din;
        out1 = reg_din;
        out2 = din;
    end
endmodule
```



```
module blocking_nonblocking(
    input din,
    input clk,
    output reg out1,
    output reg out2
);

    reg reg_din;

    always@(posedge clk) begin
        reg_din <= din;
        out1 <= reg_din;
        out2 <= din;
    end
endmodule
```



组合逻辑使用非阻塞赋值，时序逻辑使用阻塞赋值的现象？

Verilog 语法知识



■ 再谈always

- 用于描述电路的功能块，功能可以很复杂，也可以很简单
- 一个module中可以有很多个always块，相互之间独立工作，没有先后顺序之分
- 根据敏感信号列表中的内容有两种模式：

(1) 敏感信号列表中为*，或者电平信号（例如：a,b,c）：描述的是组合逻辑电路

建议使用*号，避免漏掉敏感信号产生不必要的错误

(2) 敏感信号列表中为边沿信号（例如：posedge/negedge clk）：描述的是时序电路

当always块中有多个边沿时，其中仅能有一个为时钟边沿，其他为异步复位或异步置位信号

当always块中仅有一个边沿时，其中仅能是时钟边沿，若有复位或置位一定是同步方式

• 敏感信号规则

(1) always块的敏感列表不能同时包含边沿信号（posedge/negedge）和电平信号

(2) 同一信号的双边沿禁止：同一时钟的上升沿和下降沿也不允许同时出现在敏感列表中

```
always@(posedge clk) begin
    if(rst) begin
        q <= 0;
    end
    else if(set) begin
        q <= 1;
    end
    else begin
        q <= d;
    end
end
```

```
always@(posedge clk or posedge rst or posedge set) begin
    if(rst) begin
        q <= 0;
    end
    else if(set) begin
        q <= 1;
    end
    else begin
        q <= d;
    end
end
```

```
always @(*)
    | 逻辑功能
end
```

```
//always在组合逻辑中的使用
101 always@(*)
102 begin
103
104 end
105
106 //单个沿触发的时序逻辑
107 always@(沿变化)
108 begin
109     //具体逻辑
110 end
111 //多个沿触发的时序逻辑
112 always@(<沿变化1>or<沿变化2>)
113 begin
114     //具体逻辑
115 end
```

Verilog 语法知识



■ case语句

- 与if else语句一样，属于条件语句
- if else对应优先编码电路，有优先级的
- case对应多路复用器，没有优先级

■ 使用注意事项

- 值1到值n必须各不相同
- 如果连续排列的值项执行相同语句，可以用逗号间隔值项
- default只在前面列出了case表达式的所有可能值时才可以省略，否则会在电路中产生锁存器
- case语句的表达式的值的位宽必须与值项的位宽相等，需要精确匹配

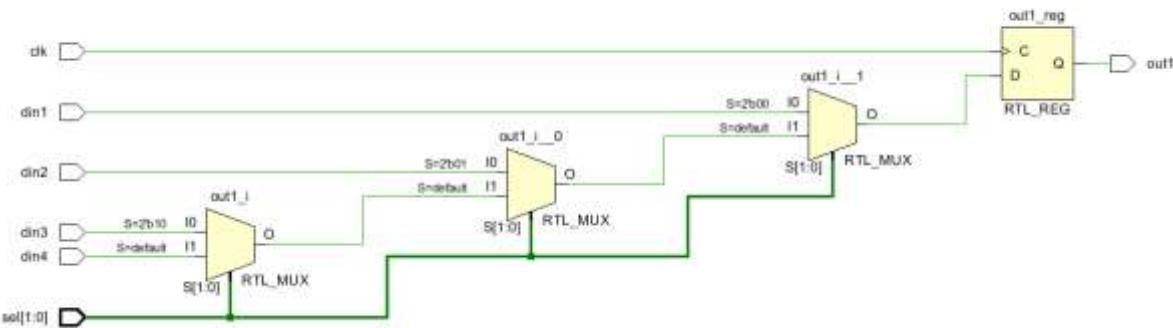
```
case (case表达式)
    表达式值1: case分支语句1;
    表达式值2: case分支语句2;
    表达式值3: case分支语句3;
    表达式值4: case分支语句4;
    default: case默认分支语句;
endcase
```

```
case (sel)
    0:           case分支语句1;
    1,2,3,4:     case分支语句2;
    5,6,7,8:     case分支语句3;
    9:           case分支语句4;
    default:     case默认分支语句;
endcase
```

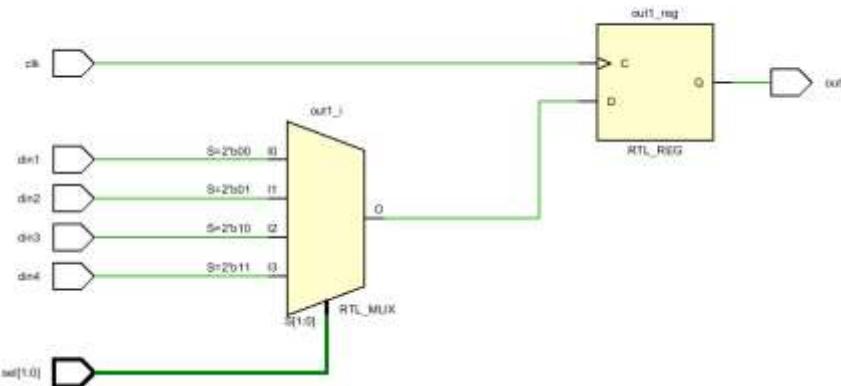
Verilog 语法知识



```
always@(posedge clk) begin
    if(sel == 2'b00)
        out1 <= din1;
    else if(sel == 2'b01)
        out1 <= din2;
    else if(sel == 2'b10)
        out1 <= din3;
    else
        out1 <= din4;
end
```



```
always @ (posedge clk ) begin
    case (sel)
        2'b00: out1 <= din1;
        2'b01: out1 <= din2;
        2'b10: out1 <= din3;
        2'b11: out1 <= din4;
        default: out1 <= din4;
    endcase
end
```



层次化程序结构



■ 层次化设计思想

- **自顶而下设计**, 从顶层开始设计, 按照系统架构逐层划分子模块 (从抽象到具体)
- **自底而上设计**, 从具体模块开始设计 (从具体到抽象)
- **优点:**
 - (1) 构建复杂电路系统的需要, 团队协作, 模块复用等
 - (2) 便于代码仿真, 降低调试难度
 - (3) 便于后期维护, 代码优化, 功能升级

注意: 硬件并行的概念, 模块使用并不是程序调用, 而是硬件的实现 (硬件复制)



层次化程序结构



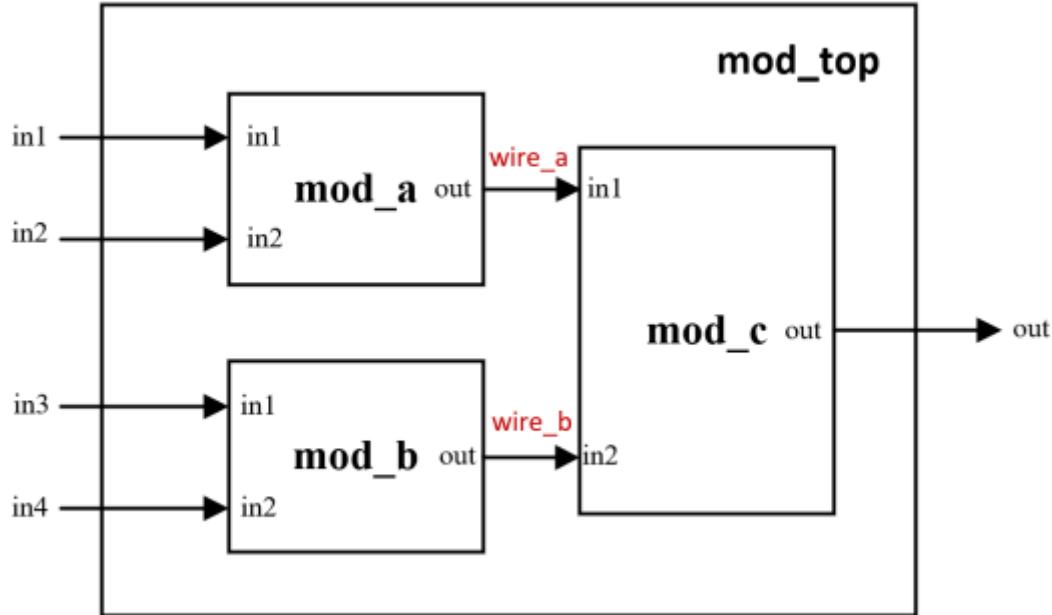
```
module mod_a(  
    input in1,  
    input in2,  
    output out  
);  
endmodule
```

创建模块
定义模块名称
定义端口名称

```
module mod_b(  
    input in1,  
    input in2,  
    output out  
);  
endmodule
```

```
module mod_top(  
    input in1,  
    input in2,  
    input in3,  
    input in4,  
    output out  
);  
endmodule
```

```
module mod_c(  
    input in1,  
    input in2,  
    output out  
);  
endmodule
```



层次化程序结构



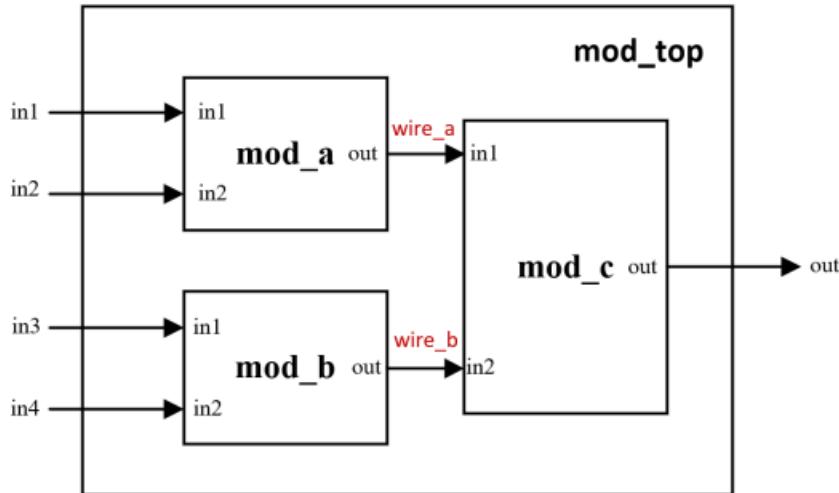
```
module mod_top(
    input in1,
    input in2,
    input in3,
    input in4,
    output out
);

wire wire_a;      定义模块间互联线
wire wire_b;

模块a例化
mod_a instance1 (.in1(in1),.in2(in2),.out(wire_a));
模块b例化
mod_b instance2 (.in1(in3),.in2(in4),.out(wire_b));
模块c例化
mod_c instance3 (.in1(wire_a),.in2(wire_b),.out(out));

endmodule
```

顶层例化



层次化程序结构



需要例化的模块中的端口名

```
module_name instance_name(.port1(port1),.port2(port2),...);
```

需要例化的模块名 例化后的模块名 例化后连接的端口名

• 例化的第一种方式

- 通过端口名字来进行例化连接
- 顺序无关，可读性强，更改容易
- 推荐使用此方式进行

https://hdlbits.01xz.net/wiki/Module_name

例化后连接的端口名

```
module_name instance_name(port1,port2,...);
```

需要例化的模块名 例化后的模块名

• 例化的第二种方式

- 通过端口位置来进行例化连接
- 顺序要严格对应

https://hdlbits.01xz.net/wiki/Module_pos



■ IP(Intelligent Property)简介

- IP核是具有知识产权核的集成电路的总称，是经过反复验证过的、具有特定功能的宏模块
- 与芯片制造工艺无关，可以移植到不同的半导体工艺中
- IP核设计已成为ASIC电路设计公司和FPGA厂商的重要任务，是其实力的体现（例如：ARM公司）
- 对FPGA开发软件来讲，其提供的ip核越丰富，用户设计就越方便
- IP核将一些在数字电路中常用但比较复杂的功能块，如FIR滤波器、SDRAM控制器、PCI接口等设计成可修改参数的模块
- CPLD/FPGA的规模越来越大，设计越来越复杂（IC的复杂度以每年55%的速率递增，而设计能力每年仅提高21%）
- 调用IP核能避免重复劳动，大大减轻工程师的负担，因此使用IP核是一个发展趋势，IP核的重用大大缩短了产品上市时间





■ IP(Intelligent Property)简介

- IP核有三种不同的存在形式：**HDL语言形式、网表形式、版图形式**，分别对应三类IP内核：软核、固核和硬核
- 软核通常是以硬件描述语言HDL源文件的形式出现
- 硬核通常是以经过完全的布局布线的网表形式提供
- 固核则是软核和硬核的折衷

MicroBlaze

■ 软核

- 软核是用VHDL/Verilog等硬件描述语言描述的功能块，不涉及用什么具体电路元件实现这些功能
- 软核只经过功能仿真，需要经过综合和布局布线才能使用
- 优点是灵活性高、可移植性强，缺点是性能并不是最优，在后续设计中存在发生错误的可能性
- 使用最为广泛



■ 硬核

- 布局和布线工艺固定，不能进行修改
- 时序控制严格，性能最优，使用难度大，不灵活

IP核使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

Synthesis and Implementation Out-of-date: 0 hours

Default Layout

File Edit Flow Tools Reports Window Layout View Help Q-Quick Access

Flow Navigator

BLOCK DESIGN design_1 project*

PROJECT MANAGER:

- Settings
- Add Sources
- Language Templates
- IP Catalog (selected)

IP INTEGRATOR

- Create Block Design
- Open Block Design
- Generate Block Design

SIMULATION

- Run Simulation

RTL ANALYSIS

- Open Elaborated Design

SYNTHESIS

- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION

- Run Implementation
- Open Implemented Design

Diagram

Search: Q

project

Nets

- two_4_input_nand_gate_0_y1
- two_4_input_nand_gate_0_y2
- tri_3_input_nor_gate_0 (74LS27 1.0)
- two_4_input_nand_gate_0 (74LS27 1.0)

System Net Properties

two_4_input_nand_gate_0_y2

Name: two_4_input_nand_gate_0_y2

Parent name: project

General Properties Pins

Tcl Console x Messages Log Reports

ENTER to select, ESC to cancel, Ctrl+Q for IP details

Design num: 1

two_4_input_nand_gate_0

XILINX

a1
b1
c1
d1
a2
b2
c2
d2

y1
y2

74LS27

tri_3_input_nor_gate_0

XILINX

a1
b1
c1
a2
b2
c2
a3
b3
c3

y1
y2
y3

74LS27

```
begin_group
start_group
create_bit_cell -type is -value xilinx.com:74ls:two_4_input_nand_gate:1.0 two_4_input_nand_gate_0
end_group
set_property location [2.0 388 -58] [get_bit_cells tri_3_input_nor_gate_0]
connect_bit_net [get_bit_pins two_4_input_nand_gate_0/y1] [get_bit_pins tri_3_input_nor_gate_0/a1]
connect_bit_net [get_bit_pins two_4_input_nand_gate_0/y2] [get_bit_pins tri_3_input_nor_gate_0/a2]
```

Type a tcl command here

IP核使用

■ Vivado集成设计环境

- IP Packager
- IP Catalog
- Block design

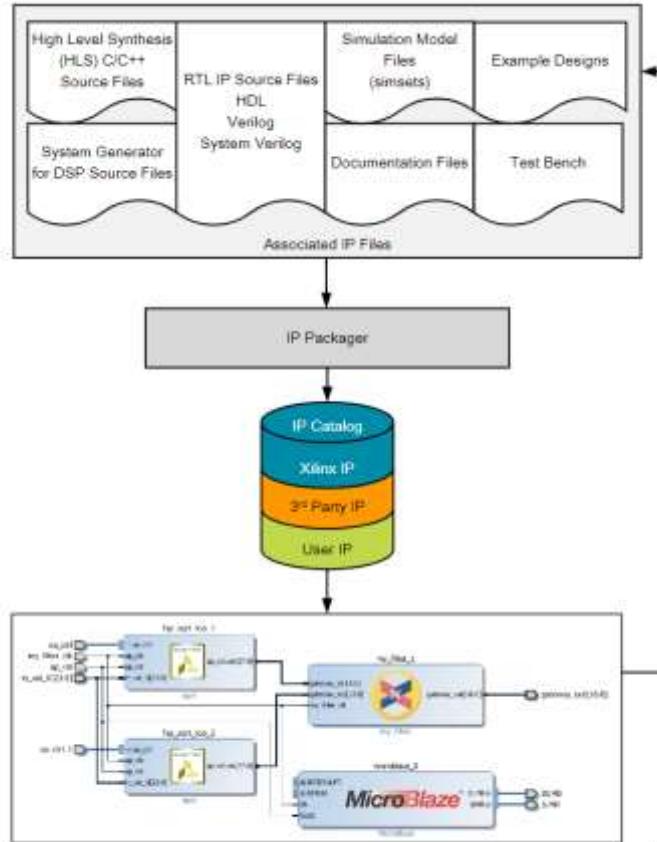
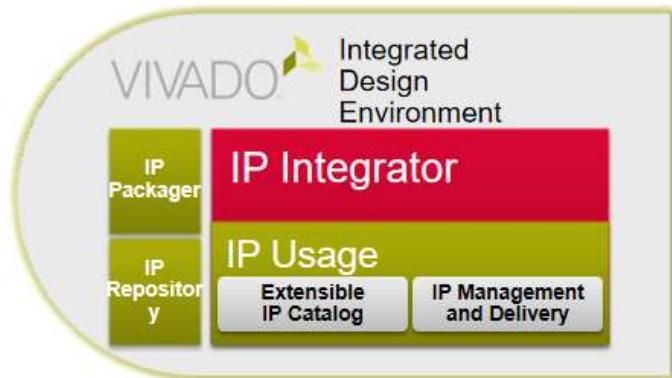


Figure 1: Vivado Design Suite IP Design Flow

IP核使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

■ IP Catalog

- IP库目录，便于快速开发
- IP均可以进行参数配置
- 绝大部分ip核免费使用，部分需要申请license

The screenshot shows the Vivado IP Catalog interface. At the top, there's a search bar and tabs for 'Project Summary' and 'IP Catalog'. Below the search bar, there are buttons for 'Cores' and 'Interfaces'. The main area displays a tree view of IP cores:

- Vivado Repository
 - Alliance Partners
 - Automotive & Industrial
 - AXI Infrastructure
 - BaseIP
 - Basic Elements
 - Communication & Networking
 - Debug & Verification
 - Digital Signal Processing
 - Embedded Processing

The screenshot shows the 'Customize IP' dialog box for the 'FIFO Generator (10.1)' component. The 'Basic' tab is selected.

Component Name: fifo_generator_0

Interface Type:

- Native AXI Memory Mapped AXI Stream

FIFO Implementation: Common Clock Block RAM

FIFO Implementation Options:

Supported Features:

	Memory Type	(1)	(2)	(3)	(4)
Common Clock (CLK)	Block RAM	✓	✓		✓
Common Clock (CLK)	Distributed RAM		✓		
Common Clock (CLK)	Shift Register				
Common Clock (CLK)	Builtin FIFO		✓	✓	✓
Independent Clocks (RD_CLK, WR_CLK)	Block RAM	✓	✓	✓	
Independent Clocks (RD_CLK, WR_CLK)	Distributed RAM		✓		
Independent Clocks (RD_CLK, WR_CLK)	Builtin FIFO		✓	✓	

(1) Non-symmetric aspect ratio: different read and write data widths
(2) First Write Has Through

OK Cancel

IP核使用



■ 免费IP核

- Bus and bridge controllers
- Debug cores
- DMA and Timers
- Inter-processor communication
- External peripheral controller Memory and memory controller
- High-speed and low-speed communication peripherals
- Other cores

■ 付费IP核

- AXI CAN controller
- AXI USB2 device
- Video IP
- Telecoms/ Wireless IP
- 付费IP核可以试用90天

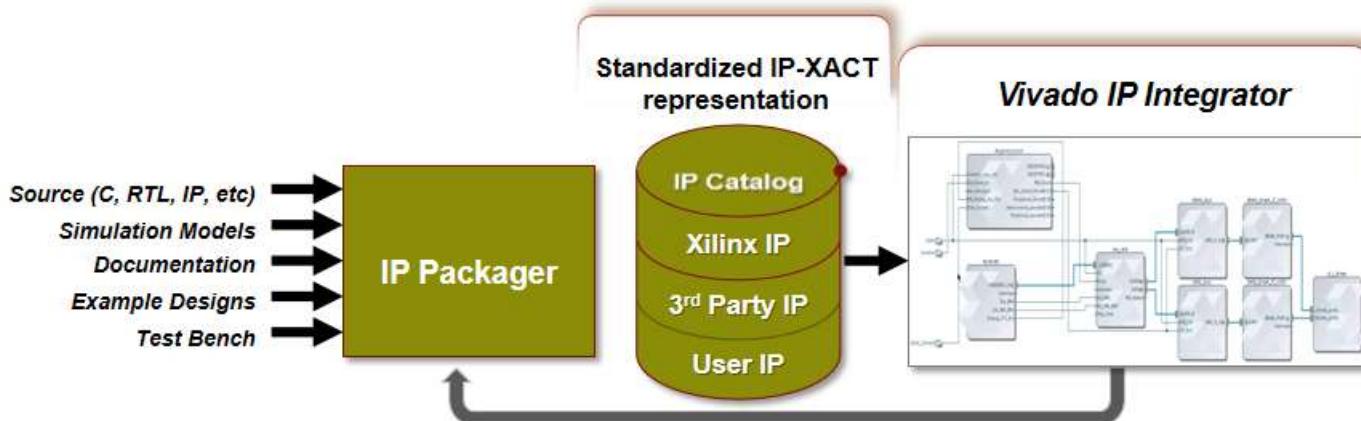
	AXI	Status	License	
● AXI UART16550	AXI4	Production	Included	xilinx.com:ip:axi_uart16550:2.0
● AXI Uartlite	AXI4	Production	Included	xilinx.com:ip:axi_uartlite:2.0
● AXI CAN	AXI4	Production	Purchase	xilinx.com:ip:can:5.0
● CANFD	AXI4	Production	Purchase	xilinx.com:ip:cantd:2.0
● I2C Bus Master Controller	AXI4	Production	Purchase	logicbricks.com:logicbricks:logi2c:0.0

IP核使用



■ IP Integrator

- 多种设计源输入，集成为IP进行标准化管理
- 多个IP可以集成为复杂IP

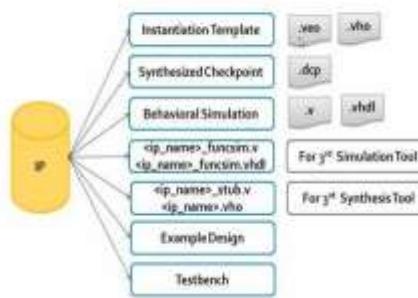




■ IP数据组成

- 例化模板: .veo(Verilog)/.vho(VHDL)
- 综合的网表文件: .dcp
- 行为级仿真文件: .v或.vhd
- 第三方仿真工具文件, 第三方综合工具文件
- 示例工程
- Testbench

IP Output Products in Vivado



IP核使用

■ 时钟IP核的使用

■ 频率综合

■ 去抖动/去偏斜

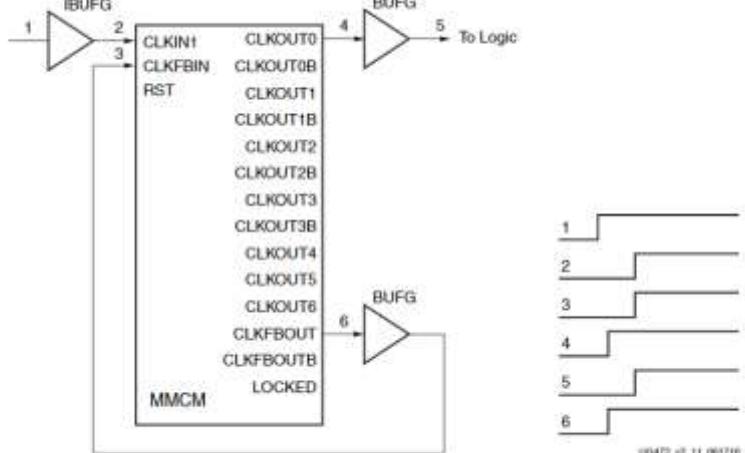
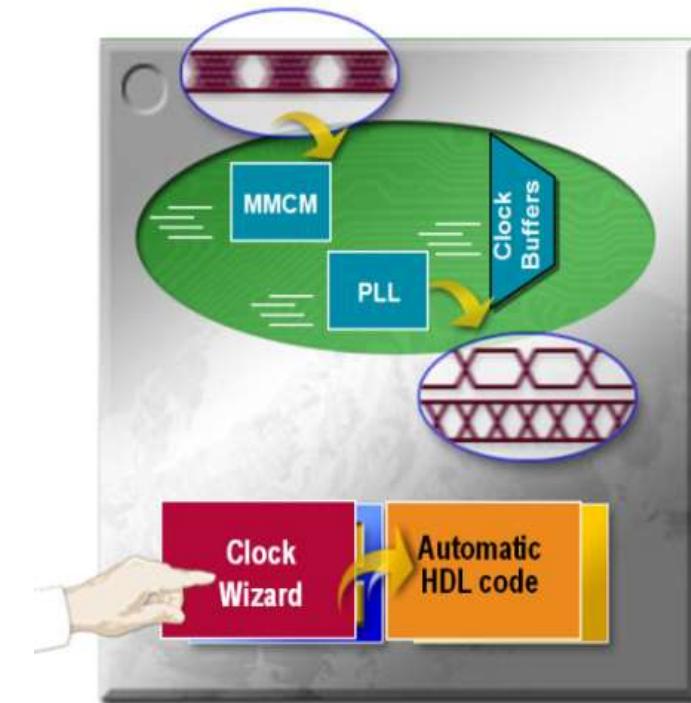


Figure 3-11: Global Clock Network Deskew Using Two BUFGs





■ 时钟IP核使用例程

- 100MHz时钟输入
- 低电平复位
- 输出1路时钟，频率为10MHz
- 带有频率锁定指示（高电平代表输出频率已锁定）
- 其余采用默认值

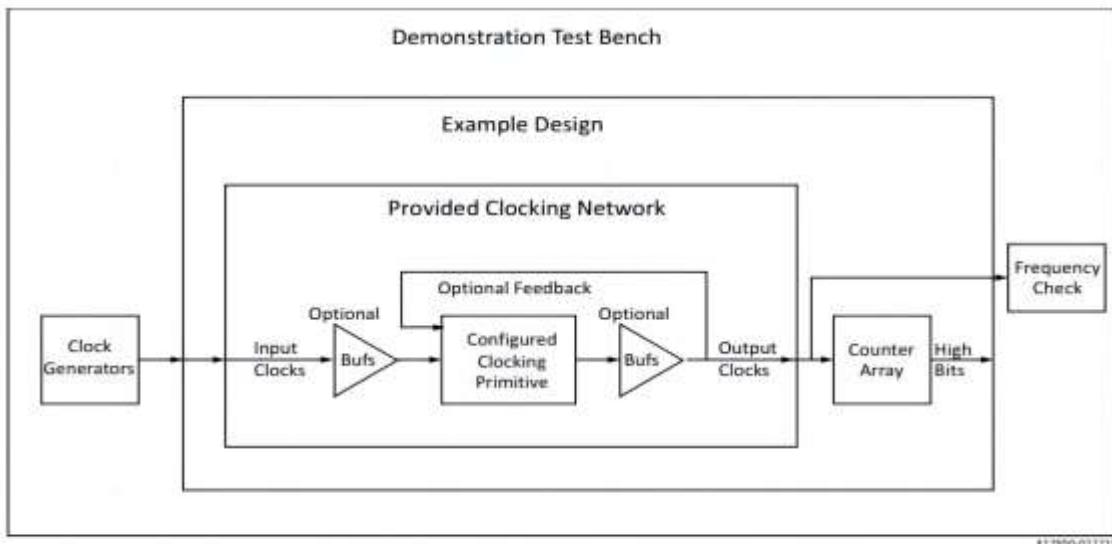
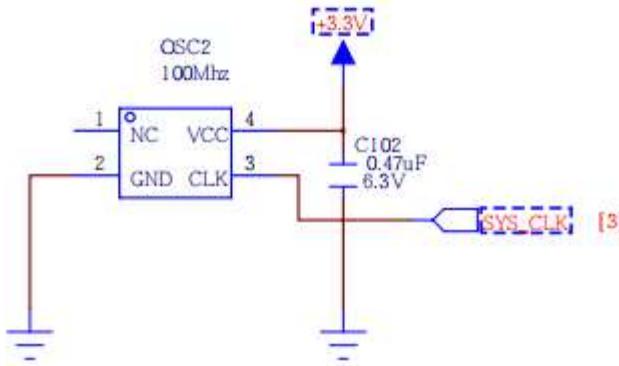


Figure 2-1: Clocking Wizard Block Diagram



■ Artix-7时钟电路

- 采用100MHz有源晶振
- 接入全局时钟网络(FPGA P17引脚)



FPGA 时钟电路

IP核使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

The screenshot shows the Vivado 2019.1 software interface with the following details:

- Project Manager:** Lab3_Seg7_Static - [C:/Xilinx/MyProject/Lab3_Seg7_Static/Lab3_Seg7_Static.xpr] - Vivado 2019.1
- Left Sidebar:**
 - PROJECT MANAGER:**
 - Settings
 - Add Sources
 - Language Templates
 - IP Catalog** (highlighted with a red box)
 - IP INTEGRATOR
 - SIMULATION
 - RTL ANALYSIS
 - SYNTHESIS
 - IMPLEMENTATION
- IP Catalog Tab:** Shows the IP Catalog interface with sections for Project Summary, IP Catalog, Cores, and Interfaces. It displays a list of available IP cores from the Vivado Repository, including SDAccel DSA Infrastructure, Network on Chip (NoC), Baseline, and Embedded Processing.
- Bottom Tab:** Design Runs (highlighted with a blue box). It lists two runs: synth_1 and impl_1, both in a 'Not started' status.
- System Tray:** Shows standard Windows icons for search, task view, file explorer, and power.
- System Bar:** Displays the date (2021/4/10), time (17:11), and language (English).

IP核使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

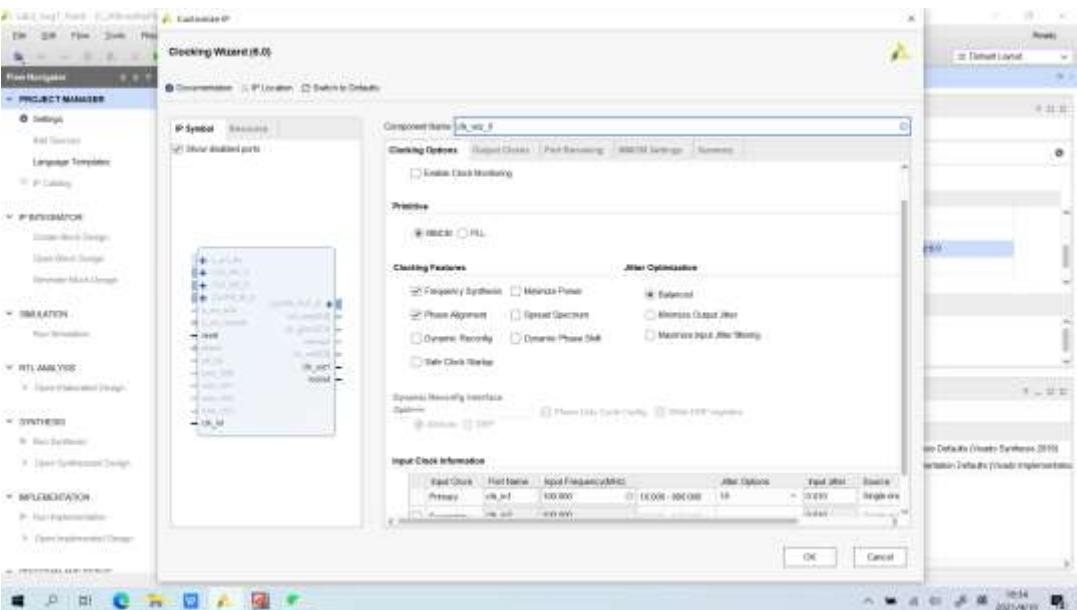
The screenshot shows the Vivado 2019.1 software interface. The top menu bar includes File, Edit, Flow, Tools, Reports, Window, Layout, View, Help, and Quick Access. The title bar indicates the project is "Lab3_Seg7_Static - [C:/Xilinx/MyProject/Lab3_Seg7_Static/Lab3_Seg7_Static.xpr] - Vivado 2019.1". The left sidebar contains sections for Project Manager, IP Integrator, Simulation, RTL Analysis, Synthesis, and Implementation. The Project Manager section is expanded, showing options like Settings, Add Sources, Language Templates, and IP Catalog. The IP Catalog section is also expanded, showing sub-options like Create Block Design, Open Block Design, and Generate Block Design. The main workspace shows the Project Manager with sources like "sim_1" and IP Properties for the "Clocking Wizard". The IP Catalog tab is selected in the Project Manager. A red box highlights the "Clocking Wizard" entry in the Catalog list, which has a detailed description: Version: 0.0 (Rev. 0), Interfaces: AXI4, Description: The Clocking Wizard creates an HDL file (Verilog or VHDL) for generating clocking logic. The Details panel shows Name: Clocking Wizard, Version: 0.0 (Rev. 0), and Interfaces: AXI4. The bottom pane shows Design Runs with two entries: synth_1 and impl_1, both with status "Not started". The system tray at the bottom right shows the date and time as 18:33, 2021/4/10.

IP核使用



- Component name: 模块名称
- Clock Monitor: 监视输入时钟状态，包括停止，故障、频率超限等
- Primitive: 源类型 (MMCM/PLL)
- Clocking Features:
 - Frequency Synthesis: 允许输出时钟与输入时钟频率不同
 - Spread Spectrum: 降低EMI
 - Phase Alignment: 相位对齐，允许输出时钟与参考源同相位
 - Minimize Power: 降低功耗
 - Dynamic Phase Shift: 允许调整输出时钟的相位
 - Dynamic Reconfiguration: 允许参数动态更改
 - Safe Clock Startup and Sequencing: 时钟安全启动和时序

1. 时钟基本信息



IP核使用



■ Jitter Optimization: 时钟抖动优化

■ input clock information: 指定输入时钟信息

2. 时钟输入信息

端口名	时钟频率						时钟输入方式: 单端或差分
	Port Name	Input Frequency(MHz)		Jitter Options	Input Jitter		
	clk_in1	100.000	×	10.000 - 800.000	UI	0.010	Single ended clock capable pin
	clk_in2	100.000		60.000 - 120.000		0.010	Single ended clock capable pin

IP核使用



- 输入输出时钟信息，名称，频率、相位、占空比等，可以选择多路输出

- 选择输出输入信号，常用的包括复位及时钟锁定输出

通常将复位信号接硬件电路的外部复位信号

将时钟锁定输出信号作为系统的全局复位信号

3. 时钟输出信息

输入输出时钟名及频率、占空比

Component Name: clk_wiz_0						
Clocking Options		Output Clocks		Port Renaming		MMC Settings
Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)
		Requested	Actual	Requested	Actual	Requested
<input checked="" type="checkbox"/> clk_out1	clk_10Mhz	10	10.000	0.000	0.000	50.000
<input type="checkbox"/> clk_out2	clk_out2	100.000	N/A	0.000	N/A	50.000

Enable Optional Inputs / Outputs for MMCM/PLL

复位信号

reset power_down input_clk_stopped

Reset Type

Active High

Active Low

锁定信号

locked clkfbstopped

EGO1外部复位信号低电平有效

IP核使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

Component Name: clk_wiz_0

Clocking Options Output Clocks Port Renaming MMCM Settings Summary

VCO Frequency
VCO Freq = 781.250 MHz

Optional Port Names

Other Pins	Port Name
locked	locked

可以更改端口名称



4. 其他信息

Component Name: clk_wiz_0

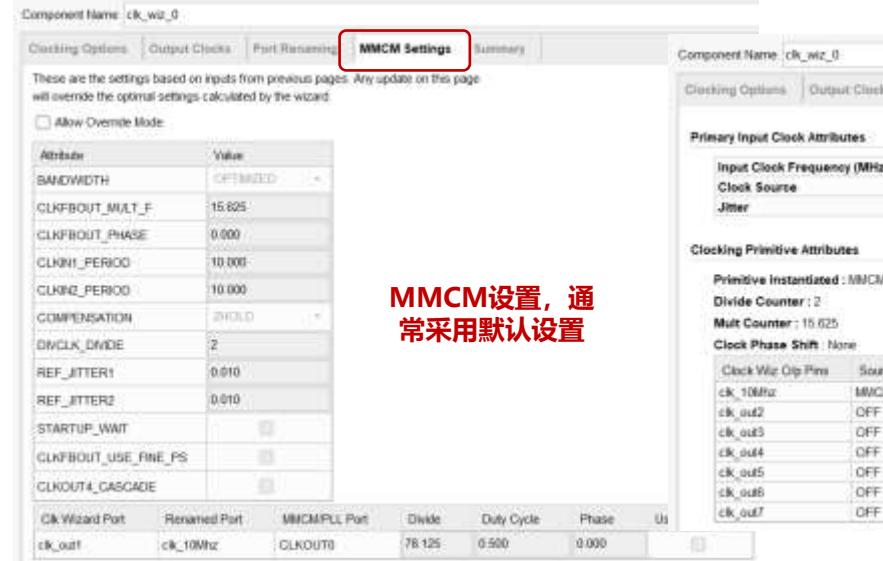
Clocking Options Output Clocks Port Renaming MMCM Settings Summary

These are the settings based on inputs from previous pages. Any update on this page will override the optimal settings calculated by the wizard.

Allow Overdrive Mode

Attribute	Value
BANDWIDTH	OPTIMIZED
CLKFBOUT_MULT_F	15.625
CLKFBOUT_PHASE	0.000
CLKIN1_PERIOD	10.000
CLKIN2_PERIOD	10.000
COMPENSATION	SWING
DIVCLK_DIVIDE	2
REF_JITTER1	0.010
REF_JITTER2	0.010
STARTUP_WAIT	
CLKFBOUT_USE_FINE_PS	
CLKOUT4_CASCADE	

Clk Wizard Port	Renamed Port	MMCM/PLL Port	Divide	Duty Cycle	Phase	Us
clk_outf	clk_10MHz	CLKOUT0	78.125	0.500	0.000	



MMC设置，通常采用默认设置

Component Name: clk_wiz_0

Clocking Options Output Clocks Port Renaming MMCM Settings Summary

Primary Input Clock Attributes

Input Clock Frequency (MHz)	100.000
Clock Source	Single-ended_clock_capable_pin
Jitter	0.010

Clocking Primitive Attributes

Primitive Instantiated : MMCM

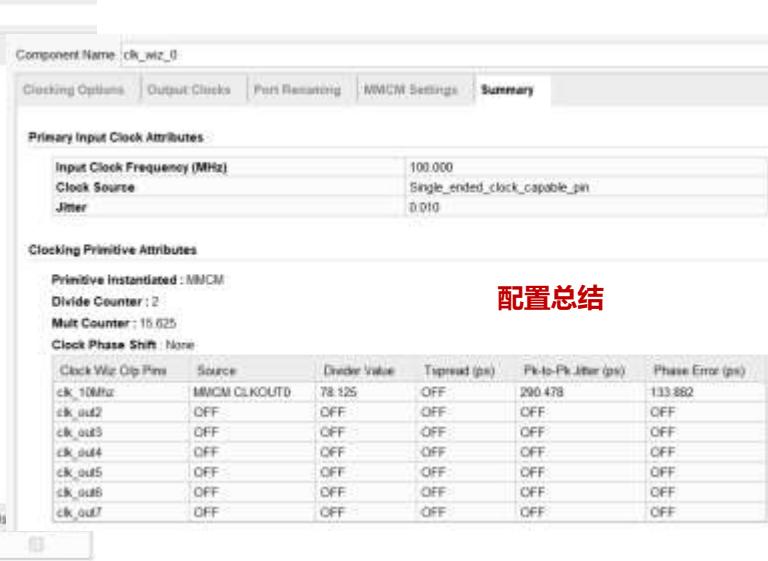
Divide Counter : 2

Mult Counter : 15.625

Clock Phase Shift : None

Clock Wiz Obj Pin	Source	Divider Value	Setup/hold (ps)	Pk-to-Pk Jitter (ps)	Phase Error (ps)
clk_10MHz	MMCM CLKOUT0	78.125	OFF	290.478	133.882
clk_out2	OFF	OFF	OFF	OFF	OFF
clk_out3	OFF	OFF	OFF	OFF	OFF
clk_out4	OFF	OFF	OFF	OFF	OFF
clk_out5	OFF	OFF	OFF	OFF	OFF
clk_out6	OFF	OFF	OFF	OFF	OFF
clk_out7	OFF	OFF	OFF	OFF	OFF

配置总结



IP Symbol Resource

1 MMCM
1 IBUFG
2 BUFG



模块资源使用情况

IP Symbol Resource

Show disabled ports

模块端口表

Port	Type	Description
clk_in1	Input	
clk_in2	Input	
clk_out0	Output	clk_10MHz
clk_out1	Output	locked
rst	Input	
reset	Input	
ref_clk	Input	
ref_jitter1	Input	
ref_jitter2	Input	
start_up	Input	
use_fine_ps	Input	
use_cascade	Input	
use_mmcm	Input	
use_ibufg	Input	
use_buflm	Input	
use_ibufg3	Input	
use_buflm3	Input	
use_ibufg4	Input	
use_buflm4	Input	
use_ibufg5	Input	
use_buflm5	Input	
use_ibufg6	Input	
use_buflm6	Input	
use_ibufg7	Input	
use_buflm7	Input	
use_ibufg8	Input	
use_buflm8	Input	
use_ibufg9	Input	
use_buflm9	Input	
use_ibufg10	Input	
use_buflm10	Input	
use_ibufg11	Input	
use_buflm11	Input	
use_ibufg12	Input	
use_buflm12	Input	
use_ibufg13	Input	
use_buflm13	Input	
use_ibufg14	Input	
use_buflm14	Input	
use_ibufg15	Input	
use_buflm15	Input	
use_ibufg16	Input	
use_buflm16	Input	
use_ibufg17	Input	
use_buflm17	Input	
use_ibufg18	Input	
use_buflm18	Input	
use_ibufg19	Input	
use_buflm19	Input	
use_ibufg20	Input	
use_buflm20	Input	
use_ibufg21	Input	
use_buflm21	Input	
use_ibufg22	Input	
use_buflm22	Input	
use_ibufg23	Input	
use_buflm23	Input	
use_ibufg24	Input	
use_buflm24	Input	
use_ibufg25	Input	
use_buflm25	Input	
use_ibufg26	Input	
use_buflm26	Input	
use_ibufg27	Input	
use_buflm27	Input	
use_ibufg28	Input	
use_buflm28	Input	
use_ibufg29	Input	
use_buflm29	Input	
use_ibufg30	Input	
use_buflm30	Input	
use_ibufg31	Input	
use_buflm31	Input	
use_ibufg32	Input	
use_buflm32	Input	
use_ibufg33	Input	
use_buflm33	Input	
use_ibufg34	Input	
use_buflm34	Input	
use_ibufg35	Input	
use_buflm35	Input	
use_ibufg36	Input	
use_buflm36	Input	
use_ibufg37	Input	
use_buflm37	Input	
use_ibufg38	Input	
use_buflm38	Input	
use_ibufg39	Input	
use_buflm39	Input	
use_ibufg40	Input	
use_buflm40	Input	
use_ibufg41	Input	
use_buflm41	Input	
use_ibufg42	Input	
use_buflm42	Input	
use_ibufg43	Input	
use_buflm43	Input	
use_ibufg44	Input	
use_buflm44	Input	
use_ibufg45	Input	
use_buflm45	Input	
use_ibufg46	Input	
use_buflm46	Input	
use_ibufg47	Input	
use_buflm47	Input	
use_ibufg48	Input	
use_buflm48	Input	
use_ibufg49	Input	
use_buflm49	Input	
use_ibufg50	Input	
use_buflm50	Input	
use_ibufg51	Input	
use_buflm51	Input	
use_ibufg52	Input	
use_buflm52	Input	
use_ibufg53	Input	
use_buflm53	Input	
use_ibufg54	Input	
use_buflm54	Input	
use_ibufg55	Input	
use_buflm55	Input	
use_ibufg56	Input	
use_buflm56	Input	
use_ibufg57	Input	
use_buflm57	Input	
use_ibufg58	Input	
use_buflm58	Input	
use_ibufg59	Input	
use_buflm59	Input	
use_ibufg60	Input	
use_buflm60	Input	
use_ibufg61	Input	
use_buflm61	Input	
use_ibufg62	Input	
use_buflm62	Input	
use_ibufg63	Input	
use_buflm63	Input	
use_ibufg64	Input	
use_buflm64	Input	
use_ibufg65	Input	
use_buflm65	Input	
use_ibufg66	Input	
use_buflm66	Input	
use_ibufg67	Input	
use_buflm67	Input	
use_ibufg68	Input	
use_buflm68	Input	
use_ibufg69	Input	
use_buflm69	Input	
use_ibufg70	Input	
use_buflm70	Input	
use_ibufg71	Input	
use_buflm71	Input	
use_ibufg72	Input	
use_buflm72	Input	
use_ibufg73	Input	
use_buflm73	Input	
use_ibufg74	Input	
use_buflm74	Input	
use_ibufg75	Input	
use_buflm75	Input	
use_ibufg76	Input	
use_buflm76	Input	
use_ibufg77	Input	
use_buflm77	Input	
use_ibufg78	Input	
use_buflm78	Input	
use_ibufg79	Input	
use_buflm79	Input	
use_ibufg80	Input	
use_buflm80	Input	
use_ibufg81	Input	
use_buflm81	Input	
use_ibufg82	Input	
use_buflm82	Input	
use_ibufg83	Input	
use_buflm83	Input	
use_ibufg84	Input	
use_buflm84	Input	
use_ibufg85	Input	
use_buflm85	Input	
use_ibufg86	Input	
use_buflm86	Input	
use_ibufg87	Input	
use_buflm87	Input	
use_ibufg88	Input	
use_buflm88	Input	
use_ibufg89	Input	
use_buflm89	Input	
use_ibufg90	Input	
use_buflm90	Input	
use_ibufg91	Input	
use_buflm91	Input	
use_ibufg92	Input	
use_buflm92	Input	
use_ibufg93	Input	
use_buflm93	Input	
use_ibufg94	Input	
use_buflm94	Input	
use_ibufg95	Input	
use_buflm95	Input	
use_ibufg96	Input	
use_buflm96	Input	
use_ibufg97	Input	
use_buflm97	Input	
use_ibufg98	Input	
use_buflm98	Input	
use_ibufg99	Input	
use_buflm99	Input	
use_ibufg100	Input	
use_buflm100	Input	
use_ibufg101	Input	
use_buflm101	Input	
use_ibufg102	Input	
use_buflm102	Input	
use_ibufg103	Input	
use_buflm103	Input	
use_ibufg104	Input	
use_buflm104	Input	
use_ibufg105	Input	
use_buflm105	Input	
use_ibufg106	Input	
use_buflm106	Input	
use_ibufg107	Input	
use_buflm107	Input	
use_ibufg108	Input	
use_buflm108	Input	
use_ibufg109	Input	
use_buflm109	Input	
use_ibufg110	Input	
use_buflm110	Input	
use_ibufg111	Input	
use_buflm111	Input	
use_ibufg112	Input	
use_buflm112	Input	
use_ibufg113	Input	
use_buflm113	Input	
use_ibufg114	Input	
use_buflm114	Input	
use_ibufg115	Input	
use_buflm115	Input	
use_ibufg116	Input	
use_buflm116	Input	
use_ibufg117	Input	
use_buflm117	Input	
use_ibufg118	Input	
use_buflm118	Input	
use_ibufg119	Input	
use_buflm119	Input	
use_ibufg120	Input	
use_buflm120	Input	
use_ibufg121	Input	
use_buflm121	Input	
use_ibufg122	Input	
use_buflm122	Input	
use_ibufg123	Input	
use_buflm123	Input	
use_ibufg124	Input	
use_buflm124	Input	
use_ibufg125	Input	
use_buflm125	Input	
use_ibufg126	Input	
use_buflm126	Input	
use_ibufg127	Input	
use_buflm127	Input	
use_ibufg128	Input	
use_buflm128	Input	
use_ibufg129	Input	
use_buflm129	Input	
use_ibufg130	Input	
use_buflm130	Input	
use_ibufg131	Input	
use_buflm131	Input	
use_ibufg132	Input	
use_buflm132	Input	
use_ibufg133	Input	
use_buflm133	Input	
use_ibufg134	Input	
use_buflm134	Input	
use_ibufg135	Input	
use_buflm135	Input	
use_ibufg136	Input	
use_buflm136	Input	
use_ibufg137	Input	
use_buflm137	Input	
use_ibufg138	Input	
use_buflm138	Input	
use_ibufg139	Input	
use_buflm139	Input	
use_ibufg140	Input	
use_buflm140	Input	
use_ibufg141	Input	
use_buflm141	Input	
use_ibufg142	Input	
use_buflm142	Input	
use_ibufg143	Input	
use_buflm143	Input	
use_ibufg144	Input	
use_buflm144	Input	
use_ibufg145	Input	
use_buflm145	Input	
use_ibufg146	Input	
use_buflm146	Input	
use_ibufg147	Input	
use_buflm147	Input	
use_ibufg148	Input	
use_buflm148	Input	
use_ibufg149	Input	
use_buflm149	Input	
use_ibufg150	Input	
use_buflm150	Input	
use_ibufg151	Input	
use_buflm151	Input	
use_ibufg152	Input	
use_buflm152	Input	
use_ibufg153	Input	
use_buflm153	Input	
use_ibufg154	Input	
use_buflm154	Input	
use_ibufg155	Input	
use_buflm155	Input	
use_ibufg156	Input	
use_buflm156	Input	
use_ibufg157	Input	
use_buflm157	Input	
use_ibufg158	Input	
use_buflm158	Input	
use_ibufg159	Input	
use_buflm159	Input	
use_ibufg160	Input	
use_buflm160	Input	
use_ibufg161	Input	
use_buflm161	Input	
use_ibufg162	Input	
use_buflm162	Input	
use_ibufg163	Input	
use_buflm163	Input	
use_ibufg164	Input	
use_buflm164	Input	
use_ibufg165	Input	
use_buflm165	Input	
use_ibufg166	Input	
use_buflm166	Input	
use_ibufg167	Input	
use_buflm167	Input	
use_ibufg168	Input	
use_buflm168	Input	
use_ibufg169	Input	
use_buflm169	Input	
use_ibufg170	Input	
use_buflm170	Input	
use_ibufg171	Input	
use_buflm171	Input	
use_ibufg172	Input	
use_buflm172	Input	
use_ibufg173	Input	
use_buflm173	Input	
use_ibufg174	Input	
use_buflm174	Input	
use_ibufg175	Input	
use_buflm175	Input	
use_ibufg176	Input	
use_buflm176	Input	
use_ibufg177	Input	
use_buflm177	Input	
use_ibufg178	Input	
use_buflm178	Input	
use_ibufg179	Input	
use_buflm179	Input	
use_ibufg180	Input	
use_buflm180	Input	
use_ibufg181	Input	
use_buflm181	Input	
use_ibufg182	Input	
use_buflm182	Input	
use_ibufg183	Input	
use_buflm183	Input	
use_ibufg184	Input	
use_buflm184	Input	
use_ibufg185	Input	
use_buflm185	Input	
use_ibufg186	Input	
use_buflm186	Input	
use_ibufg187	Input	
use_buflm187	Input	
use_ibufg188	Input	
use_buflm188	Input	
use_ibufg189	Input	
use_buflm189	Input	
use_ibufg190	Input	
use_buflm190	Input	
use_ibufg191	Input	
use_buflm191	Input	
use_ibufg192	Input	
use_buflm192	Input	
use_ibufg193	Input	
use_buflm193	Input	
use_ibufg194	Input	
use_buflm194	Input	
use_ibufg195	Input	
use_buflm195	Input	
use_ibufg196	Input	
use_buflm196	Input	
use_ibufg197	Input	
use_buflm197	Input	
use_ibufg198	Input	
use_buflm198	Input	
use_ibufg199	Input	
use_buflm199	Input	
use_ibufg2		

IP核使用



5. 生成IP组件

Component Name: clk_wiz_0

Clocking Options Output Clocks Port Renaming MMCM Settings Summary

Primary Input Clock Attributes

Input Clock Frequency (MHz)	100.000
Clock Source	Single-ended_clock_capable_pin
Jitter	0.010

Clocking Primitive Attributes

Primitive Instantiated : MMCM
Divide Counter : 2
Mult Counter : 15.625
Clock Phase Shift : None

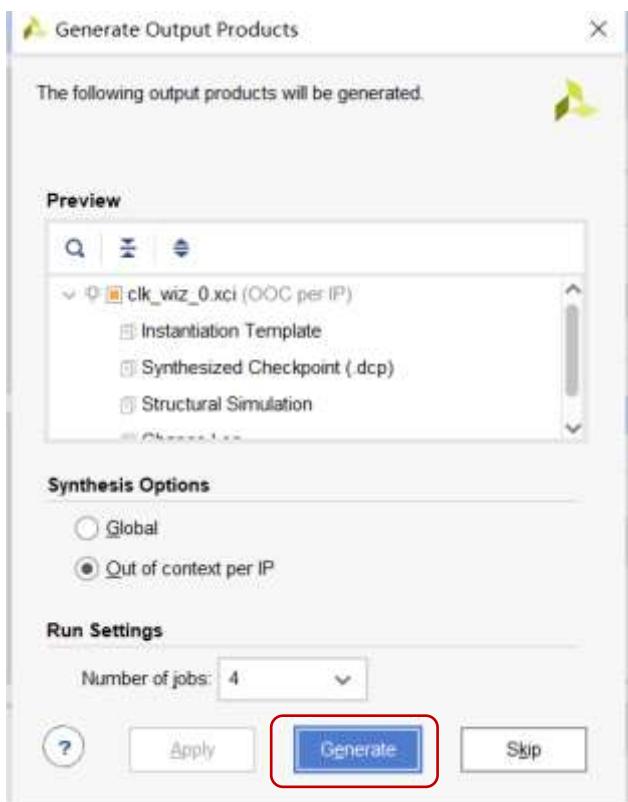
Clock Wiz O/p Pins	Source	Divider Value	Tspread (ps)	Pk-to-Pk Jitter (ps)	Phase Error (ps)
clk_10Mhz	MMCM CLKOUT0	78.125	OFF	290.478	133.882
clk_out2	OFF	OFF	OFF	OFF	OFF
clk_out3	OFF	OFF	OFF	OFF	OFF
clk_out4	OFF	OFF	OFF	OFF	OFF
clk_out5	OFF	OFF	OFF	OFF	OFF
clk_out6	OFF	OFF	OFF	OFF	OFF
clk_out7	OFF	OFF	OFF	OFF	OFF

OK Cancel

IP核使用



5. 生成IP组件



```
//----- Begin Cut here for INSTANTIATION Template ---// INST_TAG

clk_wiz_0 instance_name
(
    // Clock out ports
    .clk_10Mhz(clk_10Mhz),           // output clk_10Mhz
    // Status and control signals
    .reset(reset), // input reset
    .locked(locked), // output locked
    // Clock in ports
    .clk_in1(clk_in1));             // input clk_in1
// INST_TAG_END ----- End INSTANTIATION Template -----
```



THANKS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第四讲 (4.2 Lab2分频计数器)

2025年3月25日

Lab2分频计数器

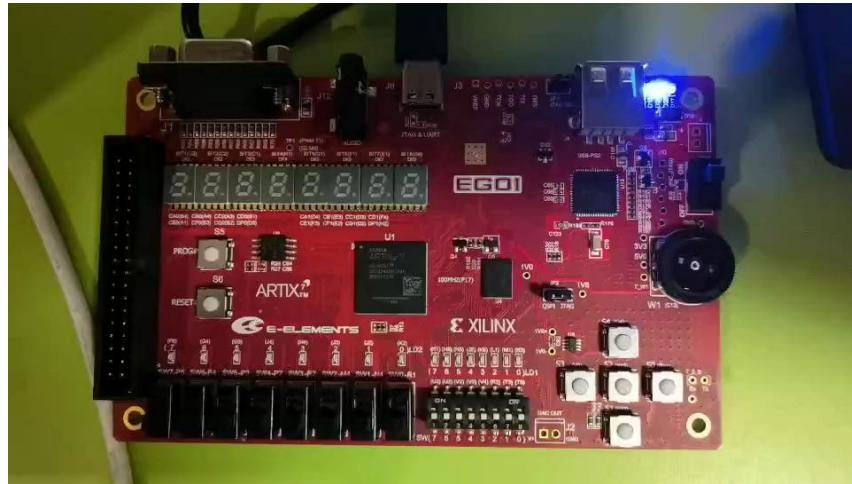


■实验内容

- 选择一颗LED，按照1Hz的频率进行闪烁
- 输入时钟为系统时钟 (100MHz)

■实验目的

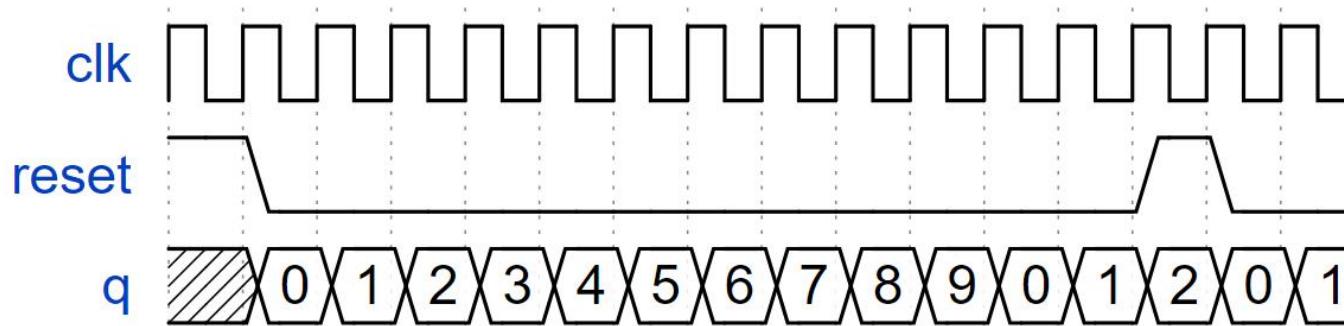
- 进一步熟悉Vivado设计工具
- 熟悉Verilog程序框架
- 学会常规计数器的设计
- 熟悉Testbench的使用





■ 计数器的功能

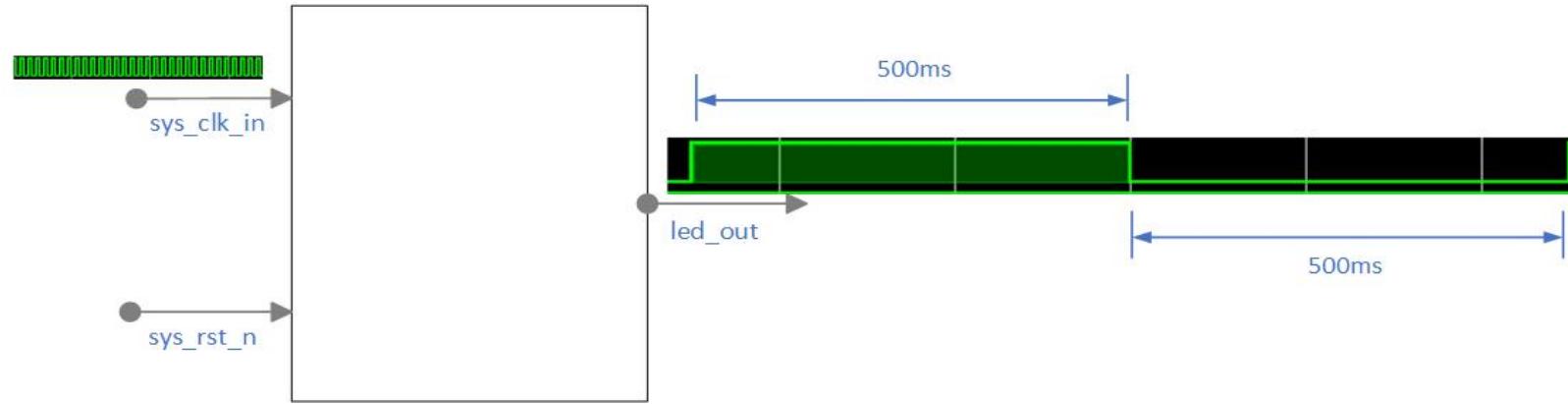
- 对输入时钟脉冲进行计数
- 可以实现分频、定时
- 可以产生序列脉冲



Lab2分频计数器



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



■ 电路结构

- 使用累加器对输入脉冲进行计数
- 复位信号有效时计数器清零
- 100MHz信号的时钟周期为10ns
- 计数器每500ms清零一次，同时将`led_out`翻转一次

$$\text{计数值} = \frac{500\text{ms}}{10\text{ns}} = 50 \times 10^6$$

Lab2分频计数器



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

序号	信号名称	输入/输出	信号描述	FPGA引脚	说明
1	sys_clk_in	输入	系统时钟输入	P17	100MHz输入
2	sys_rst_n	输入	系统复位输入	P15	低电平复位
3	led_out	输出	led控制信号输出	K3	高电平点亮

输入输出端口表



Lab2分频计数器 实验步骤

Lab2分频计数器



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 实验步骤

- 创建工程项目，工程名：Lab2_counter
- 完成设计源代码输入，文件名：Lab2_counter.v
- 完成设计约束文件输入，文件名：Lab2_counter.xdc
- 完成仿真文件输入，文件名：Lab2_counter_sim.v
- 综合 (synthesis)
- 仿真 (simulation)
- 实现 (implemetation)
- 生成bit文件
- 下载程序，查看程序运行效果

Lab2分频计数器



■输入文件 (Lab2_counter.v)

- 定义计数器变量counter，用于存放当前计数值，定义为reg类型
- 定义计数器最大值COUNTER_MAX，使用parameter便于后续修改
- 使用always模块，设计计数器模块
- 低电平异步复位 (sys_rst_n)，复位时将计数器清零、led_out清零
- 计数器在sys_clk_in上升沿时计数
- 当计数器计数到最大值时，计数器清零、led_out翻转

```
module Lab2_counter(
    input sys_clk_in,
    input sys_RST_N,
    output reg led_out
);

//定义计数器
reg [31:0] counter;

//定义计数最大值，方便后续修改
parameter COUNTER_MAX = 32'd50_000_000;

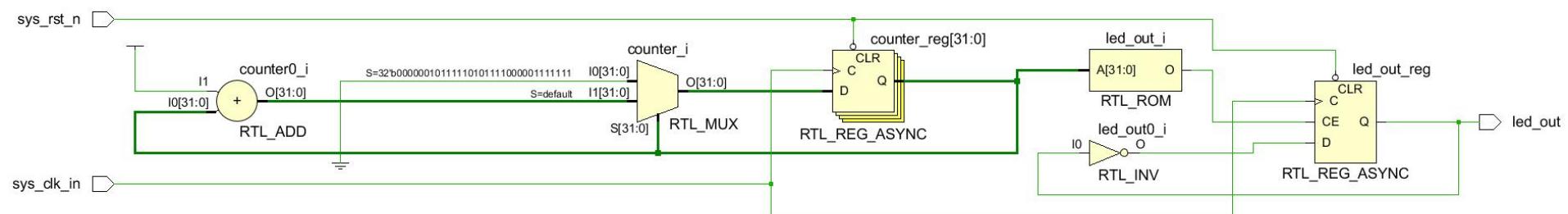
//设计一个时序电路—计数器
always @(posedge sys_clk_in or negedge sys_RST_N) begin
    //复位信号下降沿，计数器清零，led熄灭（系统复位）
    if(!sys_RST_N) begin
        counter <= 32'd0;
        led_out <= 1'b0;
    end
    // 计数到最大值时，将计数器清零，同时翻转led_out电平
    else if(counter == (COUNTER_MAX-1)) begin
        counter <= 32'd0;
        led_out <= ~led_out;
    end
    //计数器在sys_clk_in上升沿时加1计数
    else
        counter <= counter + 1'b1;
end
endmodule
```

Lab2分频计数器



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

参考RTL电路





■ 约束文件 (Lab2_counter.xdc)

- sys_clk_in配置到P17引脚
- sys_rst_n配置到P15引脚
- led_out配置到K3引脚

```
1 set_property -dict {PACKAGE_PIN P17 IO_STANDARD LVCMS33} [get_ports {sys_clk_in}]
2 set_property -dict {PACKAGE_PIN P15 IO_STANDARD LVCMS33} [get_ports {sys_rst_n} ]
3 set_property -dict {PACKAGE_PIN K3 IO_STANDARD LVCMS33} [get_ports {led_out} ]
```

Lab2分频计数器

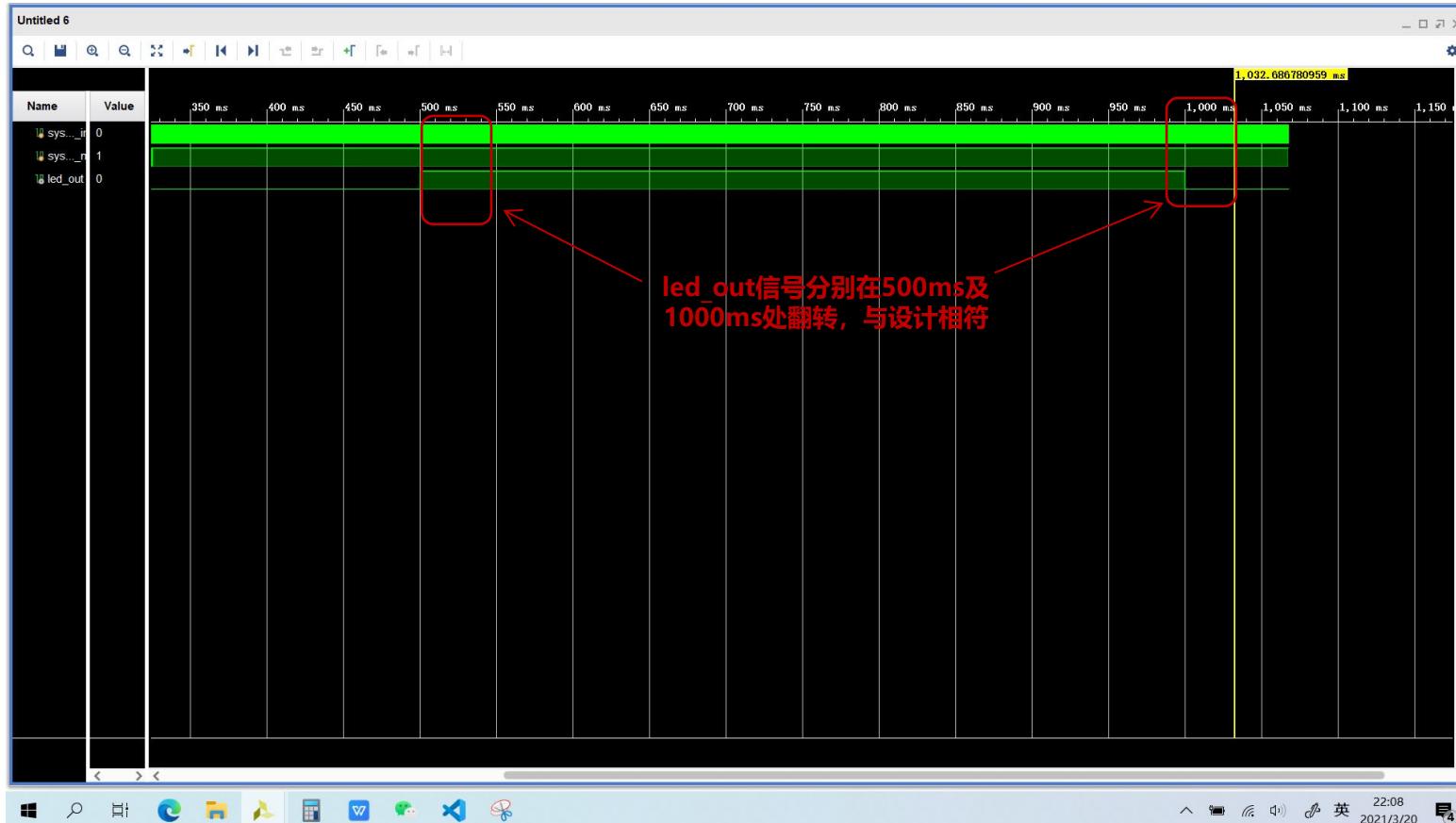


■ 仿真文件 (Lab2_counter_sim.v)

- 定义仿真输入输出信号，输入信号使用reg类型，输出信号使用wire类型
- 例化仿真测试模块
- 生成仿真测试波形

```
module Lab2_counter_sim(  
    );  
    //定义仿真输入、输出信号  
    //输入信号使用reg类型，输出信号使用wire类型  
    reg sys_clk_in;  
    reg sys_rst_n;  
    wire led_out;  
  
    //例化需要仿真的模块  
    Lab2_counter inst1  
    (  
        .sys_clk_in(sys_clk_in),  
        .sys_rst_n(sys_rst_n),  
        .led_out(led_out));  
  
    //生成仿真测试波形  
    //initial,顺序执行，至end结束  
    initial begin  
        sys_clk_in = 0;  
        sys_rst_n = 0;  
        //延迟1000ns,释放复位信号  
        #1000  
        sys_rst_n = 1;  
    end  
    //always,根据指定的时间间隔重复执行  
    always #5 sys_clk_in = ~sys_clk_in;  
  
endmodule
```

Lab2分频计数器





- 综合 (synthesis)
- 实现 (implemetation)
- 生成bit文件
- 下载程序，查看程序运行效果



1. 使用时钟IP核，输出10MHz时钟、时钟锁定信号
2. 重新设计分频计数器
3. 使用时钟IP输出的10MHz信号为主时钟，时钟锁定信号为复位信号

■ 实验步骤

- 创建工程项目，工程名：Lab2_counter_pll
- 通过IP Catalog，完成clk_10MHz IP核的配置
- 完成设计源代码输入，文件名：Lab2_counter_pll.v
- 在Lab2_counter_pll中完成clk_10MHz IP核的例化
- 完成设计约束文件输入，文件名：Lab2_counter_pll.xdc
- 完成仿真文件输入，文件名：Lab2_counter_sim_pll.v
- 综合（synthesis），仿真（simulation），实现（implementation），生成bit文件
- 下载程序，查看程序运行效果

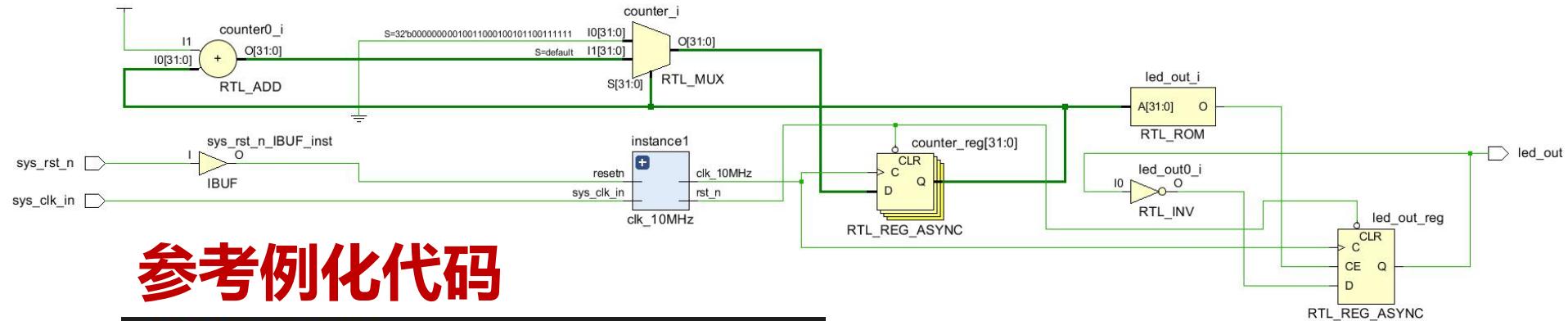
附加实验

Lab2分频计数器



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

参考RTL电路



参考例化代码

```
clk_10MHz instance1
(
    // Clock out ports
    .clk_10MHz(clk_10MHz),      // output clk_10MHz
    // Status and control signals
    .resetn(sys_rst_n), // input resetn
    .locked(rst_n),        // output locked
    // Clock in ports
    .sys_clk_in(sys_clk_in)); // input sys_clk_in
```

Lab2分频计数器



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

通过拨码开关，改变LED闪烁频率

00-->T=2s

01-->T=1s

10-->T=0.5s

11-->T=0.25s

附加实验二

要求：Lab2、附加实验一必须完成（检查确认）

课后作业：HDLbits题目



- ▶ Getting Started
- ▼ Verilog Language
 - ▶ Basics
 - ▼ Vectors
 - Vectors
 - Vectors in more detail
 - Vector part select
 - Bitwise operators
 - Four-input gates
 - Vector concatenation operator**
 - Vector reversal 1
 - Replication operator**
 - More replication
 - ▶ Modules: Hierarchy
 - ▶ Procedures
 - ▶ More Verilog Features
 - ▶ Circuits
 - ▶ Verification: Reading Simulations
 - ▶ Verification: Writing Testbenches
 - ▶ CS450

- ▶ Getting Started
- ▼ Verilog Language
 - ▶ Basics
 - ▶ Vectors
 - ▶ Modules: Hierarchy
 - ▼ Procedures
 - Always blocks (combinational)
 - Always blocks (clocked)**
 - If statement
 - If statement latches
 - Case statement
 - Priority encoder
 - Priority encoder with casez
 - Avoiding latches**
 - ▶ More Verilog Features
 - ▶ Circuits
 - ▶ Verification: Reading Simulations
 - ▶ Verification: Writing Testbenches
 - ▶ CS450

- ▶ Getting Started
- ▶ Verilog Language
- ▼ Circuits
 - ▶ Combinational Logic
 - ▼ Sequential Logic
 - ▶ Latches and Flip-Flops
 - ▼ Counters
 - Four-bit binary counter**
 - Decade counter**
 - Decade counter again**
 - Slow decade counter**
 - Counter 1-12**
 - Counter 1000**
 - 4-digit decimal counter
 - 12-hour clock
 - ▶ Shift Registers
 - ▶ More Circuits
 - ▶ Finite State Machines
 - ▶ Building Larger Circuits
 - ▶ Verification: Reading Simulations
 - ▶ Verification: Writing Testbenches
 - ▶ CS450

下节预告



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Lab3实验内容

Lab3流水灯实验

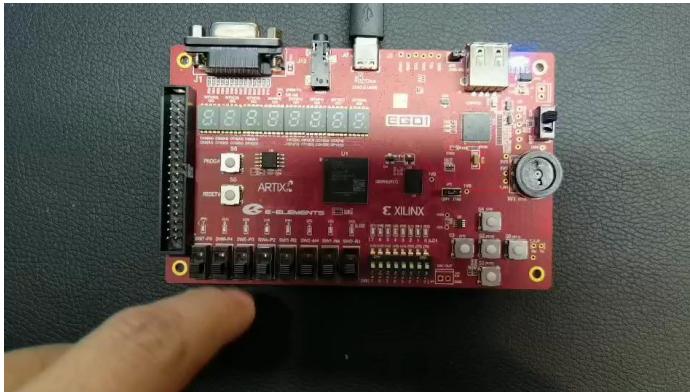
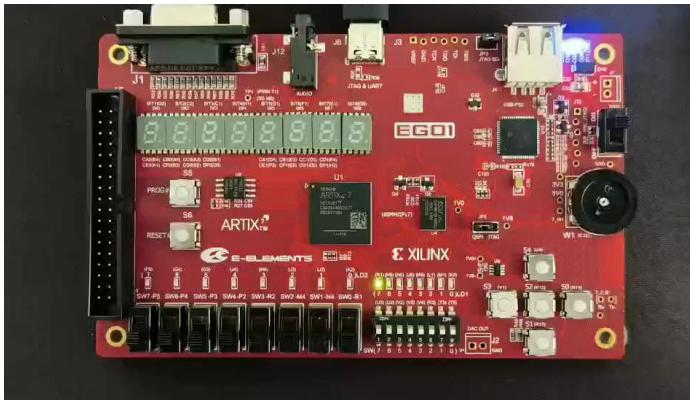


■ 实验内容

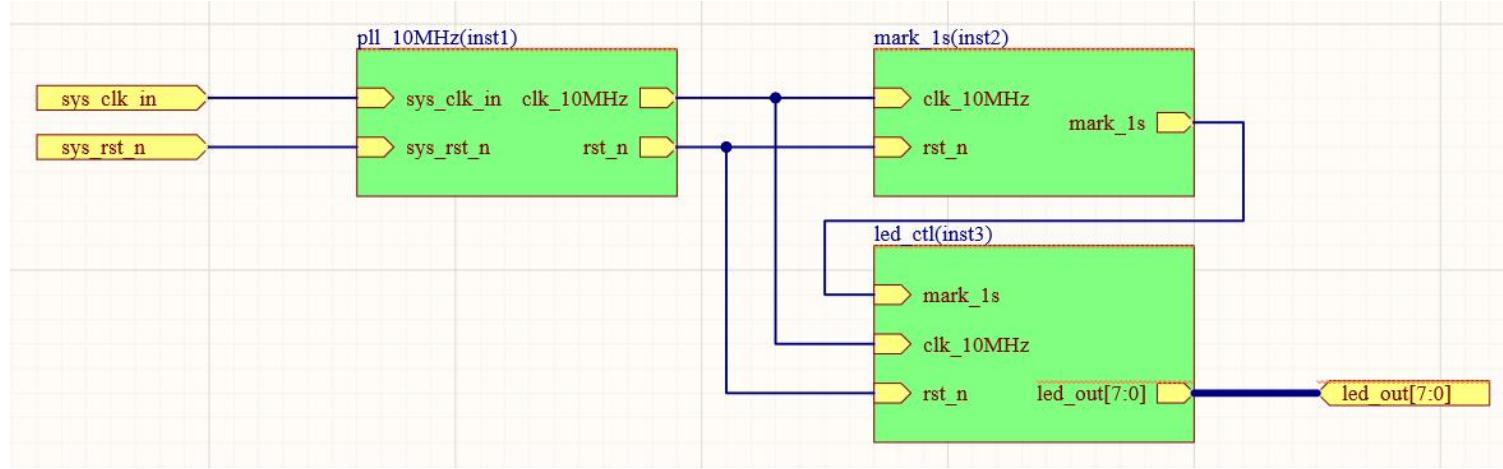
- 使用8颗LED，按照1Hz的频率进行切换点亮
- 效果1：8颗LED按照顺序依次点亮，每次点亮1颗
- 效果2：花式点亮
 - (1) 8颗LED按顺序依次点亮
 - (2) 8颗LED全亮、全灭、全亮
 - (3) 8颗LED从中间的LED开始，每次熄灭2颗
- 效果3：自行设计

■ 实验目的

- 继续熟悉基于Vivado设计工具的设计流程
- 学会层次化程序设计
- 学会连接语句（{}）的使用



Lab3流水灯实验



■ 设计思路及电路结构

- 系统复位（初始化）时输出“0000_0001”信号（点亮1颗LED）
- 使用循环左移位，按照1s周期对“0000_0001”中的“1”进行移位操作
- 设计三个模块：pll模块、秒脉冲产生模块、led控制模块
- 秒脉冲模块对输入的系统时钟进行计数，输出秒脉冲信号；led控制模块实现对led的状态控制
- 模块均采用异步复位

Lab3流水灯实验



■ 秒脉冲模块

- 设计思路：对输入的10MHz时钟 ($T=100\text{ns}$) 进行计数，定义reg [31:0] counter
counter计数到10_000_000时，mark_1s输出高电平（高电平仅维持一个clk周期）
- 详细设计：变量列表：reg [31:0] counter

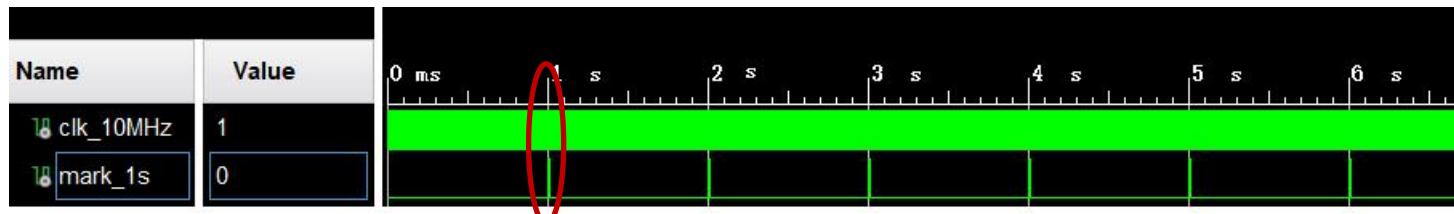
parameter COUNTER_MAX = 32'd10_000_000

功能块：always@(posedge clk_10MHz or negedge rst_n)

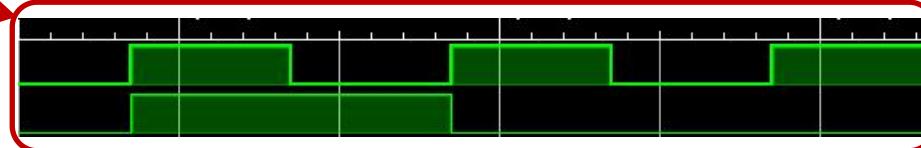
若复位信号rst_n有效，则：counter清零，mark_1s置0

否则，若counter == COUNTER_MAX-1，则：counter清零，mark_1s置1

否则，则：counter+1，mark_1s置0



$$\text{计数器清零计数值} = \frac{1\text{s}}{100\text{ns}} = 10 \times 10^6$$





THANKS



國科大杭州高等研究院
Hangzhou Institute for Advanced Study, UCAS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计

第四讲 (4.3 Flash配置文件生成与下载)

2025年3月25日

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Lab2_flash_led - [C:/Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.xpr] - Vivado 2019.1

File Edit Flow Tools Reports Window Layout View Help Quick Access

Flow Navigator SYNTHESIS
Run Synthesis
Open Synthesized Design

IMPLEMENTATION
Run Implementation
Open Implemented Design (highlighted)

Constraints Wizard
Edit Timing Constraints
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Report DRC
Report Utilization
Report Power
Schematic

PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
Open Target
Program Device
Add Configuration Memory Dev

PROJECT MANAGER - Lab2_flash_led

Sources
Design Sources (1)
Lab2_flash_led (Lab2_flash_led.v)
Constraints (1)

Hierarchy Libraries Compile Order

Properties
Lab2_flash_led.xdc
Enabled

Location: C:/Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.xdc

General Properties

需要打开实现文件

Project Summary
Overview | Dashboard

Settings Edit

Project name: Lab2_flash_led
Project location: C:/Xilinx/MyProject/Lab2_flash_led
Product family: Artix-7
Project part: xc7a35tcsg324-1
Top module name: Lab2_flash_led
Target language: Verilog
Simulator language: Mixed

1. 对bit文件进行设置

Synthesis Implementation Summary | Route Status

Status: Complete Status: Complete
Messages: 1 warning Messages: 1 warning

Tcl Console Messages Log Reports Design Runs

Name Constraints Status WNS TNS WHS THS TPWS Total Power Failed Routes LUT FF BRAMs URAM DSP Start Elapsed Run Strategy

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	synth_design Complete!								41	41	0.0	0	0	3/27/21, 11:18 AM	00:00:32	Vivado Synthesis Design
impl_1	constrs_1	write_bitstream Complete!	5.651	0.000	0.233	0.000	0.000	0.070	0	41	49	0.0	0	0	3/28/21, 9:25 PM	00:01:02	Vivado Implementation

分析和约束一个实现的设计

21:38 2021/3/28

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

The screenshot shows the Vivado 2019.1 software interface. The left sidebar has sections for Flow Navigator, Implementation, Program and Debug, and Open Hardware. A red box highlights the 'Bitstream Settings...' option under 'Open Hardware'. The main area shows the Hardware Manager with a table of hardware components and their statuses. Below it is the Configuration Memory Device Properties window. The bottom right shows the Tcl Console output of a bitstream programming operation.

1. 对bit文件进行设置

鼠标右键单击->选择Bitstream Settings

Tcl Console output:

```
Performing Program and Verify Operations...
Program/Verify Operation successful.
[0]: [labtoolstcl 44-377] Flash programming completed successfully
[0]: [program_hw.cfigmem: Time (s): cpu = 00:00:01 ; elapsed = 00:01:10 . Memory (MB): peak = 3528.125 ; gain = 0.000]
```

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

1. 对bit文件进行设置

对bit文件进行设置

The screenshot shows the Vivado 2019.1 interface. On the left, the Flow Navigator pane is open, showing the 'IMPLEMENTATION' section selected. In the center, a 'Settings' dialog box is open under the 'Bitstream' tab. The 'Configure additional bitstream settings...' button is highlighted with a red box. The 'Write Bitstream (write_bitstream)' section contains several options like 'tcl.pre', 'tcl.post', etc., each with a checkbox. On the right, a terminal window shows the command 'write_bitstream Complete' with a green checkmark. The bottom status bar shows system information: VPWS: 4.500 ns, Slack (TPWS): 0.000 ns, and other metrics.

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

1. 对bit文件进行设置

write_bitstream Complete ✓

Default Layout

Master SPI x4

选择Master SPI x4模式

OK

VIVADO

DIN

INIT_B

DOUT

Timing Summary - impl_1 (saved)

File Edit Flow Tools Reports Window Lay Settings

SYNTHESIS

IMPLEMENTATION

IMPLEMENTED DESIGN

Open Implemented Design

Run Implementation

Constraints Wizard

Edit Timing Constraints

Report Timing Summary

Report Clock Networks

Report Clock Interaction

Report Methodology

Report DRC

Report Noise

Report Utilization

Report Power

Schematic

PROGRAM AND DEBUG

Generate Bitstream

Open Hardware Manager

Open Target

Program Device

Flow Navigator

Sources Netlist

Lab2_flash_led

Nets (145)

Leaf Cells

General

Configuration

Configuration Modes

Startup

Encryption

Readback

Tcl Console

General Info

Timer Setting

Design Timing

Clock Summary

Check Timing

Intra-Clock

Inter-Clock

Help

Timing Summary - impl_1 (saved)

21:41 2021/3/28

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Lab2_flash_led - [C:/Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.xpr] - Vivado 2019.1

File Edit Flow Tools Reports Window Layout View Help Quick Access

Flow Navigator SYNTHESIS Run Synthesis Open Synthesized Design

IMPLEMENTATION Run Implementation Open Implemented Design Constraints Wizard Edit Timing Constraints Report Timing Summary Report Clock Networks Report Clock Interaction Report Methodology Report DRC Report Utilization Report Power Schematic

PROGRAM AND DEBUG Generate Bitstream Open Hardware Manager Open Target Program Device Add Configuration Memory Device

PROJECT MANAGER - Lab2_flash_led Sources Design Sources (1) Lab2_flash_led (Lab2_flash_led.v) Constraints (1) constrs_1 (1) Hierarchy Libraries Compile Order Properties Lab2_flash_led.xdc Enabled Location: C:/Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.xdc General Properties

2. 生成bit文件

Project Summary Overview Dashboard Settings Edit Project name: Lab2_flash_led Project location: C:/Xilinx/MyProject/Lab2_flash_led Project family: Artix-7 Project part: xc7a35tcsg324-1 Top module name: Lab2_flash_led Target language: Verilog Simulator language: Mixed

Synthesis Implementation Status: Complete Status: Complete Messages: 1 warning Messages: 1 warning

Tcl Console Messages Log Reports Design Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	synth_design Complete!								41	41	0.0	0	0	3/27/21, 11:18 AM	00:00:32	Vivado Synthesis Design
impl_1	constrs_1	write_bitstream Complete!	5.651	0.000	0.233	0.000	0.000	0.070	0	41	49	0.0	0	0	3/28/21, 9:25 PM	00:01:02	Vivado Implementation

分析和约束一个实现的设计

21:38 2021/3/28 英



配置Flash使用

3. 生成Flash配置文件

生成Flash配置文件

The screenshot shows the Vivado 2019.1 software interface. The 'Tools' menu is open, and the 'Generate Memory Configuration File...' option is highlighted with a red box. The main workspace displays a device map for a g324-1 device, showing four quadrants (X0Y2, X1Y2, X0Y1, X1Y0) with various logic blocks and interconnects. Below the device map, the 'Timing' tab of the 'Design Timing Summary' window is active, showing timing constraints and results. The status bar at the bottom right indicates 'write_bitstream Complete'.

Lab2_flash_led - [C:/Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.xpr] - Vivado 2019.1

File Edit Flow Reports Window Layout View Help Quick Access

Tools

Flow Navigator

IMPLEMENTATION

Open Implementation

Constraints Wizard

Edit Timing Constraints

Report Timing

Report Clock

Report Clock

Report Methodology

Report DRC

Report Noise

Report Utilization

Report Power

Schematic

Generate Memory Configuration File...

Compile Simulation Libraries...

Download Latest Boards...

Xilinx Tcl Store...

Custom Commands

Language Templates

Settings...

PROGRAM AND DEB

Generate Bitstream

Open Hardware Manager

Open Target

Program Device

Add Configuration Memory Device

Timing Summary - impl_1 (saved)

Timing

Setup Hold Pulse Width

Worst Negative Slack (WNS): 5.651 ns Total Negative Slack (TNS): 0.000 ns Number of Failing Endpoints: 0 Total Number of Endpoints: 33

Worst Hold Slack (WHS): 0.233 ns Total Hold Slack (THS): 0.000 ns Number of Failing Endpoints: 0 Total Number of Endpoints: 33

Worst Pulse Width Slack (WPWS): 4.500 ns Total Pulse Width Negative Slack (TPWS): 0.000 ns Number of Failing Endpoints: 0 Total Number of Endpoints: 34

All user specified timing constraints are met.

工具

21:45 2021/3/28

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

3. 生成Flash配置文件

(1) 选择配置器件

(2) 打开配置器件列表

(3) 搜索n25q64

Flow Navigator

IMPLEMENTATION

- Run Synthesis
- Open Synthesized Design
- Open Implemented Design
 - Constraints Wizard
 - Edit Timing Constraints
 - Report Timing Summary
 - Report Clock Networks
 - Report Clock Interaction
 - Report Methodology
 - Report DRC
 - Report Noise
 - Report Utilization
 - Report Power
 - Schematic
- PROGRAM AND DEBUG
 - Generate Bitstream
 - Open Hardware Manager
 - Open Target
 - Program Device
 - Add Configuration Memory Device

生成配置内存文件

File Edit Flow Tools Reports

Format: MCS

Memory Part: Custom Memory Size (MB): 2

Filename: ...

Options

Interface Filter

Manufacturer: All Type: All

Density (Mb): All Width: All

Search: n25q64

Name	Part	Manufac...	Alias	Family	Type	Density ...	Width
n25q64-1.8v-spi-x1_x2_x4	n25q64-1.8v	Micron		n25q	spi	64	x1_x2_x4
n25q64-3.3v-spi-x1_x2_x4	n25q64-3.3v	Micron		n25q	spi	64	x1_x2_x4

Command: write_cfgmem -format mcs -size 2 -interface SMAPx8

OK Cancel

write_bitstream Complete ✓

Default Layout

500 ns 000 ns

21:47 2021/3/28

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

3. 生成Flash配置文件

(1) 打开Flash配置文件列表

(2) 输入Flash配置文件名称
(名字可以自定义, 建议放到工程目录的如下位置: .runs->impl_1)

The screenshot shows the Xilinx Vivado IDE interface. On the left, the 'Implementation' tab is selected in the 'Flow Navigator'. In the center, a 'Write Memory Configuration File' dialog is open, showing options for 'Format' (MCS), 'Memory Part' (n25q64-3.3v-spi-x1_x2_x4), and 'Custom Memory Size (MB)' (8). Below it, a 'Specify Configuration Filename' dialog is open, with 'Save In:' set to 'impl_1'. A red box highlights the 'File name:' input field and the 'Save' button. The system tray at the bottom right shows the date and time as 2021/3/28 21:54.

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

3. 生成Flash配置文件

(1) 选择SPIx4接口

(2) 加载bit文件

(3) 打开bit文件列表

(4) 选择bit文件

File Edit Flow Tools Reports

Flow Navigator Run Synthesis

IMPLEMENTATION

- Run Implementation
- Open Implemented Design
 - Constraints Wizard
 - Edit Timing Constraints
 - Report Timing Summary
 - Report Clock Networks
 - Report Clock Interaction
 - Report Methodology
 - Report DRC
 - Report Noise
 - Report Utilization
 - Report Power
 - Schematic
- PROGRAM AND DEBUG
 - Generate Bitstream
 - Open Hardware Manager
 - Open Target
 - Program Device
 - Add Configuration Memory Device

生成配置内存文件

Format: MCS

Memory Part: n25q64-3.3v-spi-x1_x2_x4

Custom Memory Size (MB): 8

Filename: C:/Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.runs/impl_1/Lab2_flash_led.mcs

Options

Interface: SPIx4

Load bitstream files

Start address: 00000000 Direction: up Bitfile: C:/Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.runs/impl_1/Lab2_flash_led.bit

Specify Datafile Filename

Look in: impl_1

.Xil

Lab2_flash_led.bit

Recent Directories

File: Lab2_flash_led.bit

Directory: C:/Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.runs/impl_1

Created: Today at 21:44 PM

Accessed: Today at 21:44 PM

Modified: Today at 21:44 PM

Size: 2.1 MB

Type: Bitsream file

Owner: LAPTOP-8ARIPGC2/mzpan

write_bitstream Complete ✓

Default Layout

22:03 2021/3/28

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

3. 生成Flash配置文件

Create a configuration file to program the device

Format: MCS

Memory Part: n25q64-3.3v-spi-x1_x2_x4

Custom Memory Size (MB): 8

Filename:

Options

Interface: SPIx1

Load bitstream files Daisy chain configuration file

Start address: 00000000 Direction: up Bitfile: C:/Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.runs/impl_1/Lab2_flash_led.bit

Load data files

Start address: 00000000 Direction: up Datafile:

Write checksum
 Disable bit swapping
 Overwrite

(1) 选择参数

Command: mat mcs -size 8 -interface SPIx1 -loadbit {up 0x00000000 "C:/Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.runs/impl_1/Lab2_flash_led.bit"} -checksum -enablebitswap

(2) 确定

OK Cancel

write_bitstream Complete ✓

Default Layout

500 ns
000 ns

21:53 2021/3/28

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

3. 生成Flash配置文件

The screenshot shows the Vivado 2019.1 software interface. On the left, the Flow Navigator pane is open, showing the 'IMPLEMENTATION' section with options like 'Run Implementation' and 'Open Implemented Design'. In the center, the 'IMPLEMENTED DESIGN' window displays a device map with four memory blocks labeled X0Y2, X1Y2, X0Y0, and X1Y0. A message box in the foreground says 'Generate Memory Configuration File' with the message 'Generate memory configuration file completed successfully.' and an 'OK' button. At the bottom, the 'Timing' tab of the 'Design Timing Summary' window is active, showing timing constraints and results. The status bar at the bottom right indicates 'write_bitstream Complete' and the date '2021/3/28'.

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Lab2_flash_led - [C:/Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.xpr] - Vivado 2019.1

File Edit Flow Tools Reports Window Layout View Help Quick Access

Flow Navigator SYNTHESIS Run Synthesis Open Synthesized Design

IMPLEMENTATION Run Implementation Open Implemented Design Constraints Wizard Edit Timing Constraints Report Timing Summary Report Clock Networks Report Clock Interaction Report Methodology Report DRC Report Utilization Report Power Schematic

PROGRAM AND DEBUG Generate Bitstream Open Hardware Manager Open Target Program Device Add Configuration Memory Dev

HARDWARE MANAGER - localhost/xilinx_tcf/Xilinx/1234-tulA

There are no debug cores. Program device Refresh device

Hardware

Name	Status
xc7a35t_0 (2)	Programmed
XADC (System Monitor)	
n25q64-3.3v-spi-x1_X	

Flash器件列表

(1) 进入Hardware Manager界面
(2) 连接器件
(3) 确认是否有Flash配置器件

Tcl Console Messages Serial I/O Links Serial I/O Scans

```
set_property PROGRAM.FILE {C:/Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.runs/impl_1/Lab2_flash_led.bit} [get_hw_devices xc7a35t_0]
current_hw_device [get_hw_devices xc7a35t_0]
refresh_hw_device -update_hw_probes false [lindex [get_hw_devices xc7a35t_0] 0]
INFO: [Labtools 27-1434] Device xc7a35t (JTAG device index = 0) is programmed with a design that has no supported debug core(s) in it.
create_hw_cfgmem -hw_device [get_hw_devices xc7a35t_0] -mem_dev [lindex [get_cfgmem_parts {n25q64-3.3v-spi-x1_x2_x4}] 0]
```

Type a Tcl command here

System Monitor Core: XADC

22:06 2021/3/28

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

4. 下载Flash配置文件

上一步如果没有Flash器件，需要手动添加

The screenshot shows the Vivado 2019.1 interface with the 'Hardware Manager' tab selected. In the center, the 'xc7a35t_0' device is listed under the 'Hardware' section. A context menu is open over the device node, with the 'Add Configuration Memory Device...' option highlighted. Other options in the menu include 'Hardware Device Properties...', 'Program Device...', 'Verify Device...', 'Refresh Device', 'Show Bus Plot...', 'Boot from Configuration Memory Device', 'Program BBR Key...', 'Clear BBR Key...', 'Program eFUSE Registers...', and 'Export to Spreadsheet...'. At the bottom of the window, a terminal window displays Tcl commands related to the bitstream write process.

```
set_property PROGRAM.FILE {C:/Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.runs/impl_1/Lab2_flash_led.bit} [get_hw_devices xc7a35t_0]
current_hw_device [get_hw_devices xc7a35t_0]
refresh_hw_device -update_hw_probes false [lindex [get_hw_devices xc7a35t_0] 0]
INFO: [Labtools 27-1434] Device xc7a35t (JTAG device index = 0) is programmed with a design that has no supported debug core(s) in it.
create_hw_cfgmem -hw_device [get_hw_devices xc7a35t_0] -mem_dev [lindex [get_cfgmem_parts {n25q64-3.3v-spi-x1_x2_x4}] 0]
```

Type a Tcl command here

添加配置储存器件

File Edit Flow Tools Reports Window Layout View Help Quick Access

Dashboard

Flow Navigator

SYNTHESIS

Run Synthesis

Open Synthesized Design

IMPLEMENTATION

Run Implementation

Open Implemented Design

Constraints Wizard

Edit Timing Constraints

Report Timing Summary

Report Clock Networks

Report Clock Interaction

Report Methodology

Report DRC

Report Utilization

Report Power

Schematic

PROGRAM AND DEBUG

Generate Bitstream

Open Hardware Manager

Open Target

Program Device

Add Configuration Memory Device

Default Layout

write_bitstream Complete

22:08 2021/3/28

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

4. 下载Flash配置文件

对Flash器件进行编程

Lab2_flash_led - [C:/Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.xpr] - Vivado 2019.1

File Edit Flow Tools Reports Window Layout View Help Quick Access

Flow Navigator

SYNTHESIS

- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION

- Run Implementation
- Open Implemented Design
 - Constraints Wizard
 - Edit Timing Constraints
 - Report Timing Summary
 - Report Clock Networks
 - Report Clock Interaction
 - Report Methodology
 - Report DRC
 - Report Utilization
 - Report Power
 - Schematic

PROGRAM AND DEBUG

- Generate Bitstream
- Open Hardware Manager
 - Open Target
 - Program Device
 - Add Configuration Memory Dev

HARDWARE MANAGER - localhost/xilinx_tcf/Xilinx/1234-tfA

There are no debug cores. Program device Refresh device

Hardware

Name	Status
xc7a35t_0 (2)	Programmed

Configuration Memory Dev

Name	Memory Part	Memory type
n25q64-3.3v-spi-x1_x2_x4	n25q64-3.3v-spi-x1_x2_x4	spi

Tcl Console

```
set_property PROGRAM.FILE [C:/Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.runs/impl_1/Lab2_flash_led.bit] [get_hw_devices xc7a35t_0]
current_hw_device [get_hw_devices xc7a35t_0]
refresh_hw_device -update_hw_probes false [lindex [get_hw_devices xc7a35t_0] 0]
INFO: [Labtools 27-1434] Device xc7a35t (JTAG device index = 0) is programmed with a design that has no supported debug core(s) in it.
create_hw_cfgmem -hw_device [get_hw_devices xc7a35t_0] -mem_dev [lindex [get_cfgmem_parts {n25q64-3.3v-spi-x1_x2_x4}] 0]
```

Type a Tcl command here

22:09 2021/3/28

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

4. 下载Flash配置文件

选择Flash配置文件
选择prm文件，与Flash配置文件同目录

其余设置采用默认参数

The screenshot shows the Vivado interface with the 'Program and Debug' tab selected. A 'Hardware Manager' dialog box is open, titled 'Program Configuration Memory Device'. Inside, the 'Memory Device' dropdown is set to 'n25q64-3.3v-spi-x1_x2_x4'. The 'Configuration file' field contains the path 'C:/Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.runs/impl_1/Lab2_flash_led.mcs'. The 'PRM file' field contains the path 'Xilinx/MyProject/Lab2_flash_led/Lab2_flash_led.runs/impl_1/Lab2_flash_led.prm'. Under 'Program Operations', the 'Erase' and 'Program' checkboxes are checked. The 'OK' button at the bottom right of the dialog is highlighted with a red box.

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

4. 下载Flash配置文件

擦除操作

Performing erase operation - Step 1 of 2...

INFO: [Labtools 27-3164] End of startup status: HIGH
INFO: [Labtools 27-2302] Device xc7a35t (JTAG device index = 0) is programmed with a design that has 1 SPI core(s).
program_hw_cfgmem -hw_cfgmem [get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices xc7a35t_0] 0]]
Mfg ID : 20 Memory Type : ba Memory Capacity : 17 Device ID 1 : 0 Device ID 2 : 0
Performing Erase Operation...

Type a Tcl command here

The screenshot shows the Vivado 2019.1 interface with the 'Hardware Manager' open. A progress dialog titled 'Program Configuration Memory Device' is displayed, showing 'Performing erase operation - Step 1 of 2...'. The 'Tcl Console' tab at the bottom shows log messages related to the erase operation. The 'Flow Navigator' on the left indicates the current step is 'PROGRAM AND DEBUG'.

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

4. 下载Flash配置文件

烧录操作，下载时间会略长

The screenshot shows the Vivado 2019.1 interface with the following details:

- Hardware Manager:** Shows a list of connected devices:
 - localhost (1) - Connected
 - xilinx_tcf/Xilinx/1234-tulA (Open)
 - xc7a35t_0 (2) - Programmed
- Properties Dialog:** A modal window titled "Program Configuration Memory Device" is open, showing a progress bar for "Performing program operation - Step 2 of 2..." at 48% completion.
- Tcl Console:** Displays the command history and output for the current operation, including:

```
program_hw_cfgmem -hw_cfgmem [get_property PROGRAM.HW_CFGMEM [lindex [get_hw_devices xc7a35t_0] 0]]  
Mfg ID : 20  Memory Type : ba  Memory Capacity : 17  Device ID 1 : 0  Device ID 2 : 0  
Performing Erase Operation...  
Erase Operation successful.  
Performing Program and Verify Operations...
```
- System Taskbar:** Shows standard Windows taskbar icons for Start, Search, Task View, Edge, File Explorer, and File History, along with system status icons for battery, signal, and volume.

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

4. 下载Flash配置文件

烧录完成

The screenshot shows the Vivado 2019.1 interface with the following details:

- Top Bar:** File, Edit, Flow, Tools, Reports, Window, Layout, View, Help, Quick Access.
- Flow Navigator:** SYNTESIS (Run Synthesis, Open Synthesized Design), IMPLEMENTATION (Run Implementation, Open Implemented Design, Constraints Wizard, Edit Timing Constraints, Report Timing Summary, Report Clock Networks, Report Clock Interaction, Report Methodology, Report DRC, Report Utilization, Report Power, Schematic).
- HARDWARE MANAGER:** Shows a connected device labeled "localhost (1)" and a programmed device "xilinx_tcf/Xilinx/1234-tulA (Open) xc7a35t_0 (2)".
- Properties Panel:** Shows the selected device "xc7a35t_0 (2)" as programmed.
- Program Flash Dialog:** A modal dialog box displays the message "Flash programming completed successfully." with an "OK" button.
- Tcl Console:** Shows the command "program_hw_cfgmem" and its execution details, including time, memory usage, and gain.
- Bottom Taskbar:** Includes icons for search, file operations, and system status.

配置Flash使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

**电路板重新上电
或按下S5按键重新加载配置程序**



THANKS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第五讲 (5.1 Verilog语法2)

2025年4月1日



1. 数组及存储器
2. 表达式
3. 编译器指令
4. 复位



数组及存储器

- ◆ Verilog可以使用**数组**的方式对**存储器**进行建模描述，也可以使用**IP核**

- ◆ reg [7:0] mem [0:255]: 定义了一个1维数组，数组大小256，数组元素位宽8bit

```
reg [wordsize:0] array_name [0:array_size];
```

数组名称
数组位宽
数组大小

- ◆ reg [15:0] mem[0:3] [0:255]: 定义了一个2维数组，数组大小4x256，数组元素位宽16bit

- ◆ FPGA开发设计过程中，**不推荐使用Verilog建模RAM**，建议使用例化IP核的方式进行RAM设计

- ◆ 数组使用的注意事项

- (1) 注意数组与变量类型的区别
- (2) 数组全部元素的赋值不能在一条赋值语句中完成（不支持整体赋值）
- (3) 向量型数组直接索引到数组元素的某一位：Verilog-1995不支持（需要借助中间转换），Verilog-2001之后版本支持
- (4) testbench中可以使用系统函数对数组进行初始化，IP核支持使用文件方式进行初始化

变量
赋值

```
reg [7:0] mem;  
mem [5] = 1'b0;
```

数组
赋值

```
reg [7:0] mem [0:255];  
mem [5] = 8'b1101_0011;
```

reg mem [4:0]; **数组不能**
mem = 5'b00100; **X 整体赋值**

```
reg [4:0] mem;  
mem = 5'b00100; ✓
```

reg [7:0] mem [0:255];
reg [7:0] bit_mem; **借助中**
bit_mem = mem [5]; **间转换**
ledout = bit_mem [3];
ledout = mem [5][3]; **直接**
|

Verilog 语法知识



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

The screenshot shows the Vivado 2019.1 software interface. The main window is divided into several panes:

- PROJECT MANAGER - Lab2_counter**: Shows the project structure with sources (Lab2_counter.v), constraints, simulation sources, and utility sources.
- IP Catalog**: Displays the IP Catalog search results for "mem". It lists various cores and IP blocks, including AXI4 Direct Memory Access, SDx Memory Subsystem, and Basic Elements (Memory Elements, Block Memory Generator, Distributed Memory Generator).
- Design Runs**: Shows the synthesis and implementation runs for the project. The table includes columns for Name, Constraints, Status, WNS, TNS, WHS, THS, TPWS, Total Power, Failed Routes, LUT, FF, BRAMs, URAM, DSP, Start, Elapsed, Run Strategy, and Report.

A red box highlights the "Block Memory Generator" under the Basic Elements section in the IP Catalog. A large red text overlay "块RAM/分布式RAM" is placed over the highlighted area.

Name	AXI4	Status	License	VLNV
AXI Video Direct Memory Access	AXI4, AXI4-Stream	Production	Included	xilinx.com:ip:axi_vdma:6.3
SDx Memory Subsystem	AXI4	Production	Included	xilinx.com:ip:sdx_memory_subsystem:1.0

Name	AXI4	Status	License	VLNV
Block Memory Generator		Production	Included	xilinx.com:ip:blk_mem_gen:8.4
Distributed Memory Generator		Production	Included	xilinx.com:ip:distl_mem_gen:8.0

Name	AXI4	Status	License	VLNV
AXI Direct Memory Access	AXI4, AXI4-Stream	Production	Included	xilinx.com:ip:axi_dma:7.1
AXI Multi Channel Direct Memory Access	AXI4, AXI4-Stream	Production	Included	xilinx.com:ip:axi_mcdma:1.1

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Run Strategy	Report
synth_1 (active)	constrs_1	Synthesis Out-of-date								41	33	0.0	0	0	3/15/22, 10:41 AM	00:00:25	Vivado Synthesis Defaults (Vivado Synthesis 2019)	Vivad
impl_1	constrs_1	Implementation Out-of-date	95.419	0.000	0.267	0.000	0.000	0.175		0	41	33	0.0	0	3/15/22, 10:49 AM	00:01:02	Vivado Implementation Defaults (Vivado Implementation 2019)	Vivad
clk_10MHz_synth_1	clk_10MHz	synth_design Complete!								1	0	0.0	0	0	3/15/22, 8:49 AM	00:00:28	Vivado Synthesis Defaults (Vivado Synthesis 2019)	Vivad

Verilog 语法知识



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Customize IP

Distributed Memory Generator (8.0)

Documentation IP Location Switch to Defaults

Show disabled ports

Component Name: dist_mem_gen_0

memory config Port config RST & Initialization

Options

Depth: 64 [16 - 65536]

Data Width: 16 [1 - 1024]

Memory Type

Memory Type

ROM

Single Port RAM

Simple Dual Port RAM

Dual Port RAM

a[5:0]
d[15:0]
dpra[5:0]
clk
we
spo[15:0]
i_ce
dpo[15:0]
qspo_ce
qspo_dpo_ce
qspo_dpo_clk
qspo_rst
qspo_dpo_rst
qspo_srst
qspo_dpo_srst

位宽及深度

存储器类型

分布式RAM IP核

Customize IP

Distributed Memory Generator (8.0)

Documentation IP Location Switch to Defaults

Show disabled ports

Component Name: dist_mem_gen_0

memory config Port config RST & Initialization

初始化数据文件

Load COE File

The initial memory content can be set by using a COE file. This will be passed to the core as a Memory Initialisation File (MIF).

Coefficients File: \Lab2_counter\Lab2_counter.src\resources_1\ip\mem.coe

COE Options

Default Data: 0 Radix: 16

COE File Editor - mem.coe

Key	Value
memory_initialization_radix	16
memory_initialization_vector	12 34 56 78

Validate Save Save As... Close

初始化文件编辑器

Verilog 语法知识



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Customize IP

Block Memory Generator (8.4)

Documentation IP Location Switch to Defaults

IP Symbol Power Estimation

Show disabled ports

Component Name blk_mem_gen_0

Basic Port A Options Other Options Summary

Interface Type Native Generate address interface with 32 bits

Memory Type Single PortRAM Common Clock

存储器类型

ECC Options

ECC Type No ECC

Error Injection Pins Single Bit Error Injection

Write Enable

Byte Write Enable

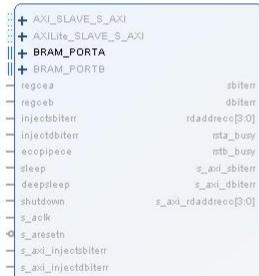
Byte Size (bits) 9

Algorithm Options

Defines the algorithm used to concatenate the block RAM primitives.
Refer datasheet for more information.

Algorithm Minimum Area

Primitive 8k2



Customize IP

Block Memory Generator (8.4)

Documentation IP Location Switch to Defaults

IP Symbol Power Estimation

Show disabled ports

Component Name blk_mem_gen_0

Basic Port A Options Other Options 位宽及深度

Memory Size

Write Width 16 Range: 1 to 4608 (bits)

Read Width 16

Write Depth 16 Range: 2 to 1048576

Read Depth 16

Operating Mode Write First Enable Port Type Use ENA Pin

Port A Optional Output Registers

Primitives Output Register Core Output Register

SoftECC Input Register REGCEA Pin

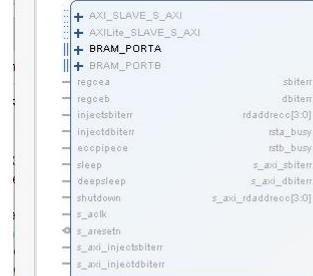
Port A Output Reset Options

RSTA Pin (set/reset pin) Output Reset Value (Hex) 0

Reset Memory Latch Reset Priority CE (Latch or Register Enable)

READ Address Change A

Read Address Change A



Verilog 语法知识



Block Memory Generator (8.4)

Documentation IP Location Switch to Defaults

IP Symbol Power Estimation

Show disabled ports

Component Name: blk_mem_gen_0

Basic Port A Options Other Options Summary

Pipeline Stages within Mux: 0 Mux Size: 1x1

初始化数据文件

Memory Initialization

Load Init File

Coe File: \Lab2_counter\Lab2_counter.srcts\sources_1\ip\mem.coe

Fill Remaining Memory Locations

Remaining Memory Locations (Hex): 0

Structural/UniSim Simulation Model Options

Defines the type of warnings and outputs are generated when a read-write or write-write collision occurs.

Collision Warnings: All

初始化文件编辑器

COE File Editor - mem.coe

Key	Value
memory_initialization_radix	16
memory_initialization_vector	12 34 56 78

Validate Save Save As... Close OK Cancel

The screenshot shows the Xilinx IP Integrator interface. On the left, there's a tree view of IP components under 'IP Catalog'. In the center, the 'Block Memory Generator' IP is selected. On the right, the 'blk_mem_gen_0' component configuration window is open. It has tabs for 'Basic', 'Port A Options', 'Other Options' (which is active), and 'Summary'. Under 'Other Options', there's a 'Memory Initialization' section with a 'Load Init File' checkbox and a 'Coe File' field containing the path '\Lab2_counter\Lab2_counter.srcts\sources_1\ip\mem.coe'. Below this is a 'Structural/UniSim Simulation Model Options' section with a 'Collision Warnings' dropdown set to 'All'. At the bottom, there's a 'COE File Editor - mem.coe' dialog box showing memory initialization parameters: 'memory_initialization_radix' set to '16' and 'memory_initialization_vector' set to '12 34 56 78'. The entire 'Memory Initialization' section and the 'COE File Editor' dialog are highlighted with red boxes. Red text annotations '初始化数据文件' and '初始化文件编辑器' are overlaid on these sections.

Verilog 语法知识



- ◆ COE文件可以通过Matlab、LabVIEW等软件生成
- ◆ IP生成完毕，在程序中例化使用
- ◆ 详细使用说明可以参考IP使用手册

```
mem.coe - 记事本
文件 编辑 查看
        基数, 2, 10或16
memory_initialization radix=16;
memory_initialization vector=12,34,56,78;

初始化数据
空格或逗号隔开
不允许负数
```

er.xpr] - Vivado 2019.1

View Help Q: Quick Access

Customize IP

Block Memory Generator (8.4)

Documentation IP Location Switch to Defaults

Product Guide (highlighted)

Change Log User Estimation

Product Webpage Reports

Answer Records



表达式

```
counter <= counter + 16'd100;  
  
if(counter == (COUNTER_MAX-1))  
  
    led_out <= ~led_out;  
  
    led_out <= led[5] | led[3];
```

- ◆ 表达式由操作数和运算符组成
- ◆ 操作数：常数 (8'h12)、参数(parameter)、线网 (wire) 、变量 (reg) 、位选及部分位选 (led[0]、led[5:3]) 、存储器和数组元素 (cnt[3])
- ◆ 运算符：算数运算符、关系运算符、逻辑运算符、按位运算符、移位运算符、赋值运算符、缩减运算符、拼接及复制运算符等
- ◆ 优先级：运算符带有优先级，建议使用括号取消优先级

Verilog 语法知识



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

操作符名称	操作符符号	操作符含义
算数运算符	+	加法
	-	减法
	*	乘法
	/	除法
	%	取模
关系运算符	>	大于
	<	小于
	>=	不小于
	<=	不大于
	==	逻辑相等
	!=	逻辑不等
逻辑运算符	&&	逻辑与
		逻辑或
	!	逻辑非
赋值运算符	=	阻塞赋值
	<=	非阻塞赋值

常用的运算符

操作符名称	操作符符号	操作符含义
按位逻辑运算符	~	按位取反
	&	按位与
		按位或
	^	按位异或
	~^或^~	按位同或
移位运算	>>	逻辑右移
	<<	逻辑左移
	>>>	算数右移
	<<<	算数左移
缩减运算符	&	缩减与
		缩减或
	^	缩减异或
	~^或^~	缩减同或
	? :	条件判断
拼接运算符	{}	数据拼接
复制运算符	{()}	数据复制



算数运算符

- ◆ 算数运算符: +, -, *, /, %
- ◆ 整数除法会截断小数部分, 只取整数, 例如: $7/4=1$
- ◆ 取模操作求出与第一个操作数符号相同的余数, 例如: $7\%4=3$, $-7\%4=-3$
- ◆ 算数表达式的位宽由最大操作数位宽决定, 建议指定位宽并匹配
- ◆ 负数以补码形式存在, 使用signed标注
- ◆ 表达式中同时有signed与unsigned类型需要格外小心, 只要有一个操作数是无符号数, 其他操作数均会被转为无符号数



关系运算符

- ◆ 关系运算符: >, <, >=, <=, ==, !=
- ◆ 关系运算的结果为真 (1) 或假 (0)
- ◆ 若表达式中操作数位宽不同, 且操作数都是无符号数, 则位宽较小的操作数在高位补零
- ◆ 若表达式中操作数位宽不同, 且操作数都是有符号数, 则位宽较小的操作数在高位补符号位
- ◆ 表达式中同时有signed与unsigned类型需要格外小心, 只要有一个操作数是无符号数, 其他操作数均会被转为无符号数



逻辑运算符

- ◆ 逻辑运算符: `&&`, `||`, `!`
- ◆ 逻辑运算的结果为真 (1) 或假 (0)
- ◆ 对于向量操作数, 非零向量即为真 (1)

```
counter1 = 'b0110;
counter2 = 'b0100;

counter1 && counter2 结果为真
counter1 || counter2 结果为真
!counter1           结果为假
```



按位逻辑运算符

- ◆ 按位逻辑运算符: ~, &, |, ^, ~^或^~
- ◆ 按位逻辑运算对输入的操作数进行逐位操作, 产生向量结果
- ◆ 若表达式中操作数位宽不同, 且操作数都是无符号数, 则位宽较小的操作数在高位补零
- ◆ 若表达式中操作数位宽不同, 且操作数都是有符号数, 则位宽较小的操作数在高位补符号位
- ◆ 表达式中同时有signed与unsigned类型需要格外小心, 只要有一个操作数是无符号数, 其他操作数均会被转为无符号数

```
counter1 = 'b0110;  
counter2 = 'b0100;
```

counter1 && counter2 结果为真
counter1 || counter2 结果为真
!counter1 结果为假

```
counter1 = 'b0110;  
counter2 = 'b0100;
```

counter1 & counter2 结果为 'b0100
counter1 | counter2 结果为 'b0110
~counter1 结果为 'b1001



移位运算符

- ◆ 移位运算符: <<, >>, <<<, >>>
- ◆ 移位运算符将操作符左侧的操作数向左或向右移位, 移位的次数由操作符右侧的操作数表示
- ◆ 对于逻辑移位(<<, >>), 由于移位而腾空的位填0
- ◆ 对于算数移位(<<<, >>>), 由于左移位而腾空的位填0, 右移位而腾空的位, 无符号数填0, 有符号数填符号位
- ◆ 一次移位相当于做 $\times 2$ 或/2操作

```
counter1 = 'b0110;
counter2 = 'b0100;

counter1 << 2 结果为 'b1000
counter1 >> 2 结果为 'b0001
```



缩减运算符

- ◆ 缩减运算符: &, |, ^, ~^或^~
- ◆ 缩减操作符对单一操作数上的所有位进行操作, 产生1位的操作结果
- ◆ 缩减与 (&) : 只要操作数有任意位置为0, 则结果为0, 否则结果为1
- ◆ 缩减或 (|) : 只要操作数有任意位置为1, 则结果为1, 否则结果为0
- ◆ 缩减异或 (^) : 只要操作数有偶数个1, 则结果为0, 否则结果为1
- ◆ 缩减同或 (~^) : 只要操作数有奇数个1, 则结果为1, 否则结果为0

```
counter1 = 'b0110;  
  
&counter1    结果为1'b0  
|counter1    结果为1'b1  
^counter1    结果为1'b0  
~~counter1   结果为1'b1
```



拼接运算符 复制运算符

- ◆ 拼接运算符: {}
- ◆ 复制运算符: {{}}
- ◆ **拼接运算符:** 将拼接运算符内部的**小独立表达式**拼接起来, 组成一个大表达式
- ◆ 拼接运算符中**不允许出现未指定位宽的值**
- ◆ **复制运算符:** 通过指定**重复次数**来实现**数据封装**

```
wire [7:0] bus_a;
wire [7:0] bus_b;
wire [7:0] bus_out;

assign bus_out[7:4] = {bus_a[3],bus_a[2],bus_b[2],bus[0]};

assign bus_out = {bus_a[7:4],bus_b[3:0]};
```

```
wire [7:0] bus_a;
wire [7:0] bus_b;

bus_a = {8{1'b1}};           bus_a结果为8'b1111_1111
bus_b = {4{1'b0,1'b1}};      bus_b结果为8'b0101_0101
```

<https://hdlbits.01xz.net/wiki/Vector4>
<https://hdlbits.01xz.net/wiki/Vector5>



常用编译器指令

- ◆ 文本替换宏命令: `define, `undef
- ◆ 条件编译宏命令: `ifdef, `ifndef, `else, `elseif, `endif
- ◆ 包含文件宏命令: `include
- ◆ 编译指令复位宏命令: `resetall
- ◆ 编译器时间宏命令: `timescale
- ◆ 编译器指令语句不需要加分号



文本替换宏命令

- ◆ 文本替换宏命令: `define, `undef
- ◆ `define类似于C语言中的#define命令
- ◆ 与parameter语句的区别, `define规定的宏定义可以跨多个文件使用
- ◆ 使用`undef取消宏定义

```
`define COUNTER_MAX 32'd1234_5678
`define WORD 16

`undef COUNTER_MAX
`undef WORD
```



条件编译宏命令

- ◆ 条件编译宏命令: `ifdef, `ifndef, `else, `elseif, `endif
- ◆ `ifdef: 如果发现宏定义, 则执行
- ◆ `ifndef: 如果未宏定义, 则执行
- ◆ `else, `elseif, `endif与`ifdef, `ifndef配合使用

```
`define WINDOWS_X64

`ifdef WINDOWS_X64
    parameter WORD_SIZE = 64;
`else
    parameter WINDOWS_X64 = 32;
`endif
```

```
`define SIMULATION

`ifdef SIMULATION
    parameter COUNTER_MAX = 32'd50_000;
`else
    parameter COUNTER_MAX = 32'd50_000_000;
`endif
```



编译指令复位宏命令

编译器时间宏命令

- ◆ 编译指令复位宏命令: `resetall
- ◆ `resetall: 将所有的编译指令重新设置为缺省值
- ◆ 编译器时间宏命令: `timescale
- ◆ `timescale: 设置编译器时间单位及时间精度
- ◆ `timescale在仿真器中有效

```
`timescale 1ns/1ps
```



FPGA中的复位

Verilog 语法知识



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

White Paper: Xilinx FPGAs



WP272 (v1.0.1) March 7, 2008

复位

Get Smart About Reset: Think Local, Not Global

By: Ken Chapman

数字系统设计的戒律之一是：“应该为所有的触发器设计主复位，这样测试工程师就会爱你，模拟仿真不会有不确定的初始值。”

One of the commandments of digital design states, "Thou shalt have a master reset for all flip-flops so that the test engineer will love you, and your simulations will not remain undefined for time eternal."



Get Smart About Reset: Think Local, Not Global

By: Ken Chapman

复位

One of the commandments of digital design states, "Thou shalt have a master reset for all flip-flops so that the test engineer will love you, and your simulations will not remain undefined for time eternal."

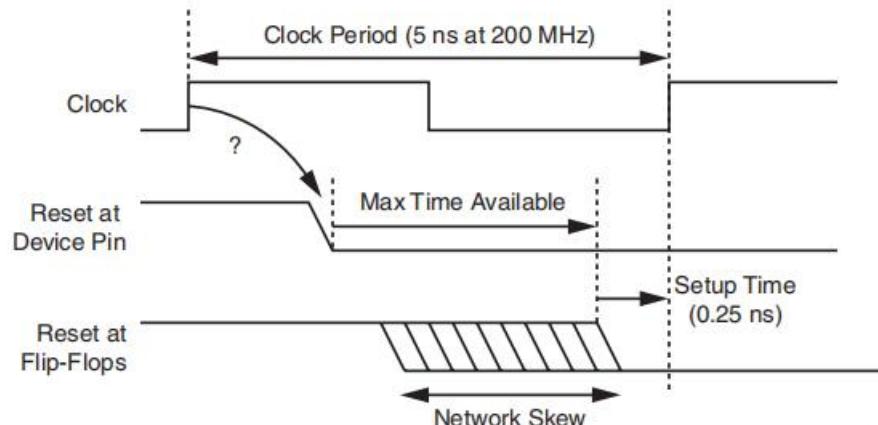
然而令人惊讶的是，对FPGA设计应用全局复位并不是一个很好的策略，应该避免使用。显然这是一个有争议的问题，因此我们需要深入探究其背后的原因。

So, some may be surprised to learn that applying a global reset to your FPGA designs is not a very good idea and should be avoided. Clearly, this is a controversial issue, so let's take a look at the reasons why such a design policy should be considered.



外部复位信号的来源

- ◆ 硬件开关或者按键
- ◆ 电源芯片产生的电源状态信号
- ◆ 专用复位芯片产生的复位信号
- ◆ 其他数字系统产生的复位信号：单片机、定时器等
- ◆ 大部分外部复位信号相对于FPGA内部触发器的时钟为异步信号
- ◆ 大部分外部复位信号为低速信号，信号周期足够完成触发器复位
- ◆ 复位信号释放的边沿是时间敏感事件，需要进行时序分析
- ◆ 全局复位信号是高扇出信号，满足所有触发器的时序要求是巨大挑战

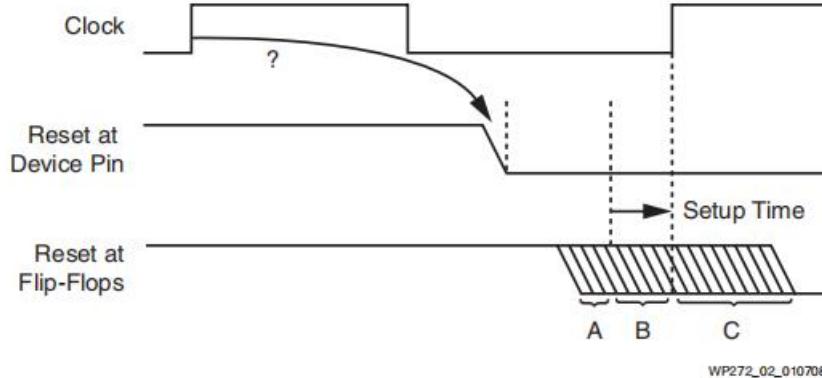


Verilog 语法知识

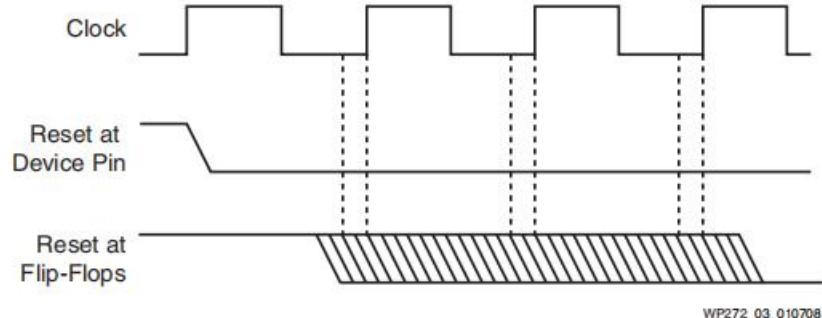


外部复位信号的时序分析

- ◆ 情况一：复位信号释放在一个时钟周期内
- ◆ 复位信号在A处释放，下一个时钟边沿，触发器可以进行有效复位
- ◆ 复位信号在B处释放，不满足建立时间要求，产生亚稳态
- ◆ 复位信号在C处释放，再下一个时钟边沿，触发器可以进行有效复位



- ◆ 情况二：复位信号释放在多个时钟周期内
- ◆ 电路规模变大时发生概率增大
- ◆ 复位信号不满足时序要求的概率变大，电路变得不可靠





Does It Really Matter?

The good news is that 99.99% of the time, the timing of the reset release really doesn't matter. With statistics like that, it isn't surprising that most circuits work. However, if you have ever had one of the circuits that doesn't work the first time, then maybe you have encountered one of the 0.01% cases and have been unlucky enough to have released the reset at the wrong time.



好消息是，在99.99%的时间里，复位信号的释放时间并不重要。有了这样的统计数据，大多数电路都能正常工作就不足为奇了。然而，如果你曾经遇到一次有一个电路不能工作，那么你可能遇到了一个0.01%的情况，不幸地在错误的时间释放了复位。



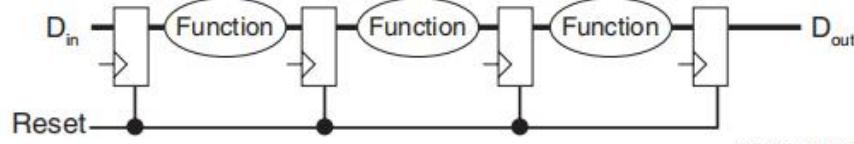
问题1：FPGA电路还需要进行复位设计吗？



复位信号的必要性

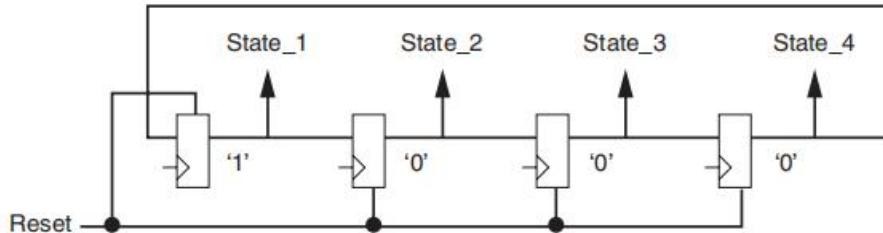
- ◆ 图4所示的流水线电路不需要复位
- ◆ 即便初始状态未知产生错误，也会在数个时钟周期后正常工作
- ◆ 图5所示的状态机需要复位
- ◆ 如果不同触发器间复位释放时间不一致，会丢失状态，产生错误

结论：没有反馈的电路不需要复位，有反馈的电路需要认真设计复位



WP272_04_010708

Figure 4: Reset for a Pipeline



WP272_05_010708

Figure 5: Reset for a One-Hot State Machine



Xilinx FPGA初始化复位

- ◆ Xilinx FPGA配置或重新配置时，所有的单元都会被初始化
- ◆ Xilinx的器件也有嵌入处理的系列，软核或硬核。在程序执行第一条指令前，程序和数据区域已经定义好了

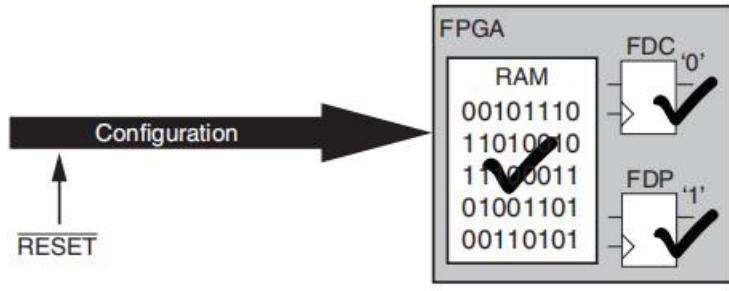


Figure 6: FPGA Configuration



异步复位，同步释放 解决0.01%的问题

◆ 通过内部生成同步复位信号的方法来复位触发器

◆ 推荐的电路形式：4级触发器级联

(1) 异步复位信号进行复位操作时，4个触发器被预设为1

(2) 异步复位信号释放后，经过4个时钟周期输出同步化的复位信号

(3) 采用多级同步的目的是降低亚稳态出现的概率

◆ FPGA内部锁相环输出的Locked信号是同步信号

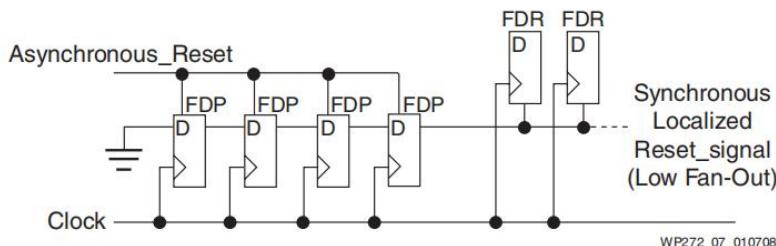


Figure 7: Localized Reset



全局复位的坏处

- ◆ 复位网络需要**占用布线资源**
- ◆ 导致其余信号的布线信号受到影响，**降低了它们布线的自由度**
- ◆ 复位网络占用大量布线资源，使得Place&Route的**时间大大增加**
- ◆ 复位信号需要**占用大量的逻辑资源**，需要使用触发器的专用复位管脚
- ◆ **额外的逻辑消耗**降低了系统的性能
- ◆ **增大整个设计的规模**





Summary

A design implemented in a Xilinx FPGA does not require insertion of a global reset network. For the vast majority of any design, the initialization state of all flip-flops and RAM following configuration is more comprehensive than any logical reset will ever be. There is no requirement to insert a reset for simulation because nothing will be undefined. Since a Xilinx FPGA is already fully tested silicon, scan logic and running test vectors are not necessary in the design. So, a global reset is not needed as part of this process either.

**应识别系统中必须真正复位的关键部分，
并且在启动或运行过程中，这些复位信号的释放必须像同步电路中其他信号一样仔细设计。**

Inserting a global reset will impact development time and final product costs even if time and cost cannot be easily quantified. With the trend towards higher speed clocks and complete systems on a chip, the reliability issues must be taken seriously. The critical parts of a system that must truly be reset should be identified and the release of those resets on start up or during operation must be controlled as carefully as any other signal within a synchronous circuit.

When creating each section of a design, simply ask, "Does this bit need to be reset"?



问题2：同步复位与异步复位的区别？

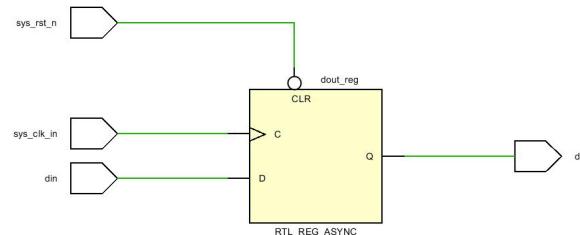
Verilog 语法知识



- ◆ 信号敏感列表中包含复位信号，会被综合为异步复位
- ◆ 异步复位：不利于时序分析，容易产生亚稳态；没有时钟信号也可复位，对复位信号周期要求低
- ◆ 信号敏感列表中不包含复位信号，会被综合为同步复位
- ◆ 同步复位：利于时序分析，降低亚稳态概率；没有时钟信号无法复位，复位信号周期要大于时钟周期

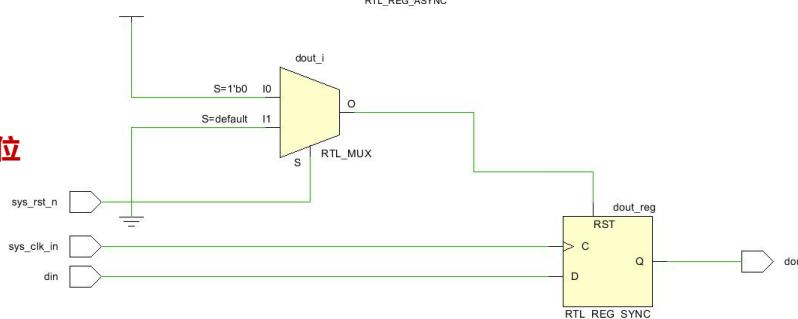
```
always@(posedge sys_clk_in or negedge sys_rst_n) begin
    if(!sys_rst_n)
        dout <= 1'b0;
    else
        dout <= din;
end
```

异步复位



```
always@(posedge sys_clk_in ) begin
    if(!sys_rst_n)
        dout <= 1'b0;
    else
        dout <= din;
end
```

同步复位



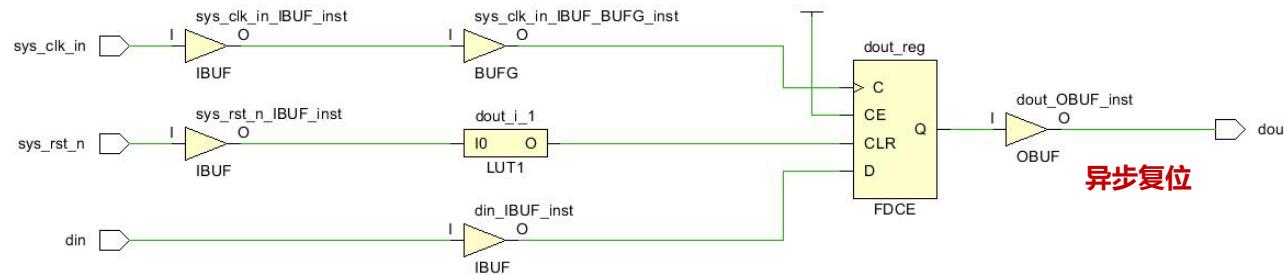
Verilog 语法知识



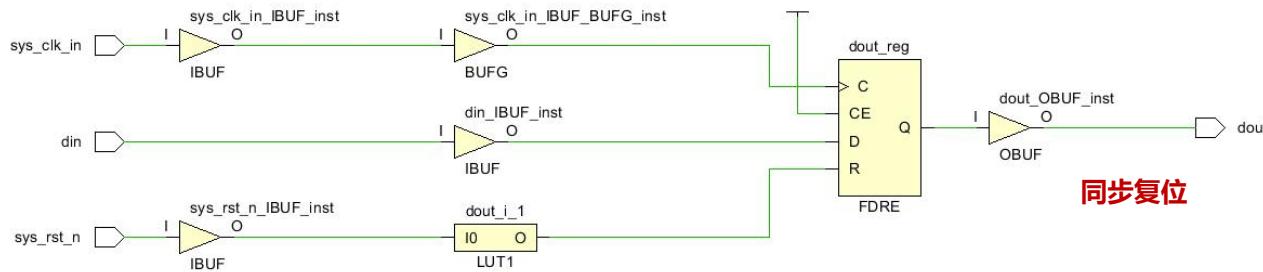
◆ Xilinx 7系列FPGA同步复位及异步复位在资源使用上没有区别

◆ 异步复位使用FDCE：带使能功能的异步清除D触发器

◆ 同步复位使用FDRE：带使能功能的同步清除D触发器



异步复位



同步复位



问题3：高电平与低电平复位的区别？

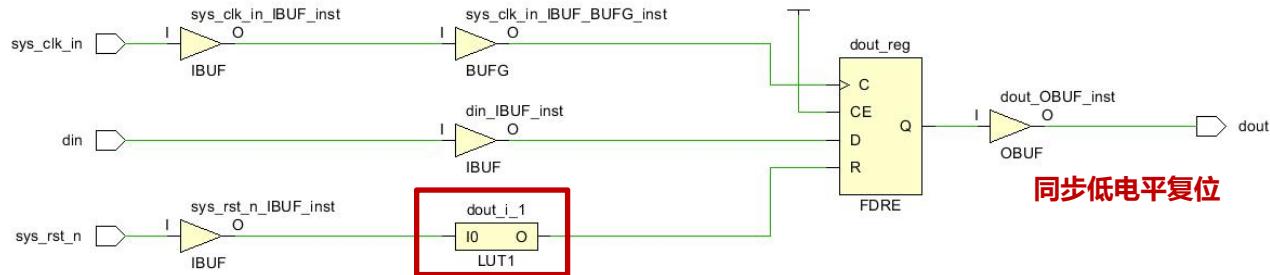
Verilog 语法知识



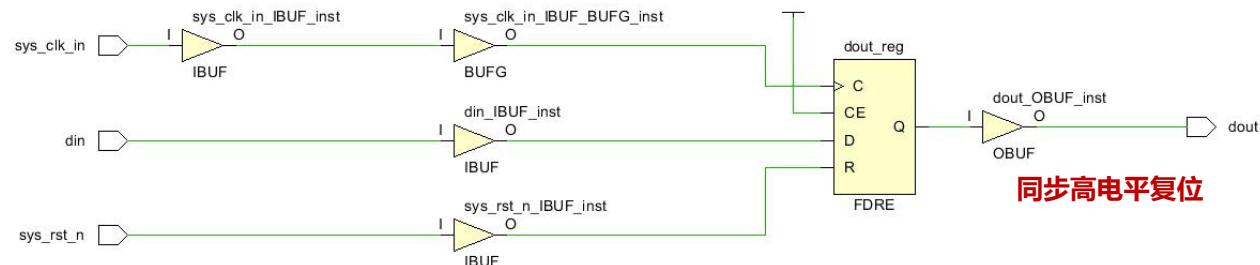
- ◆ Xilinx 7系列FPGA中的触发器使用高电平复位
- ◆ 如果设计中采用低电平复位设计，综合时会插入LUT进行取反
- ◆ 建议采用高电平复位，如果外部复位信号为低电平，建议在顶层对该复位信号进行取反后使用

```
always@(posedge sys_clk_in) begin
    if(!sys_rst_n)
        dout <= 1'b0;
    else
        dout <= din;
end
```

```
always@(posedge sys_clk_in) begin
    if(sys_rst_n)
        dout <= 1'b0;
    else
        dout <= din;
end
```



同步低电平复位



同步高电平复位



THANKS



國科大杭州高等研究院
Hangzhou Institute for Advanced Study, UCAS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS



FPGA电路软硬件设计 第五讲 (5.2 Lab3流水灯实验)

2025年4月1日

Lab3流水灯实验

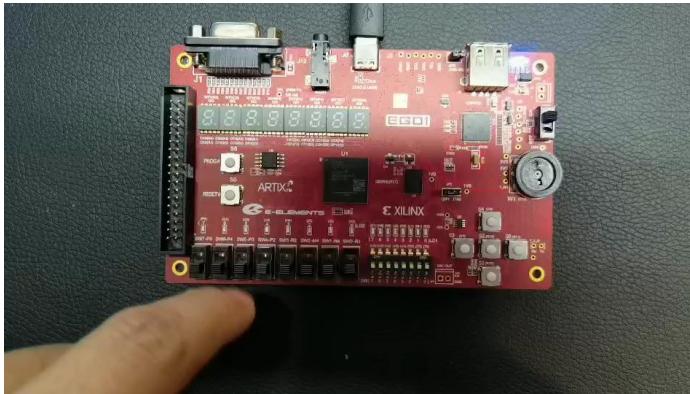
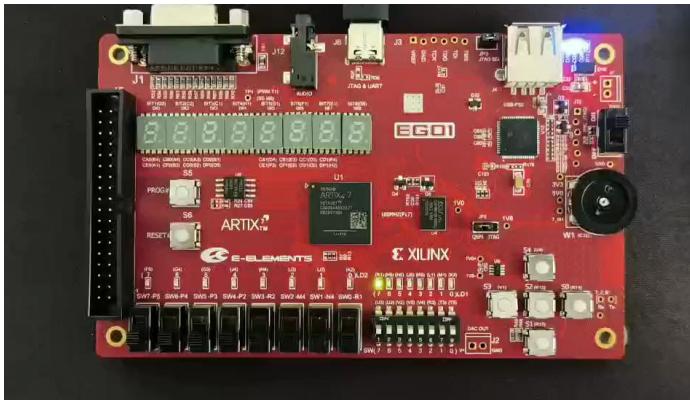


■ 实验内容

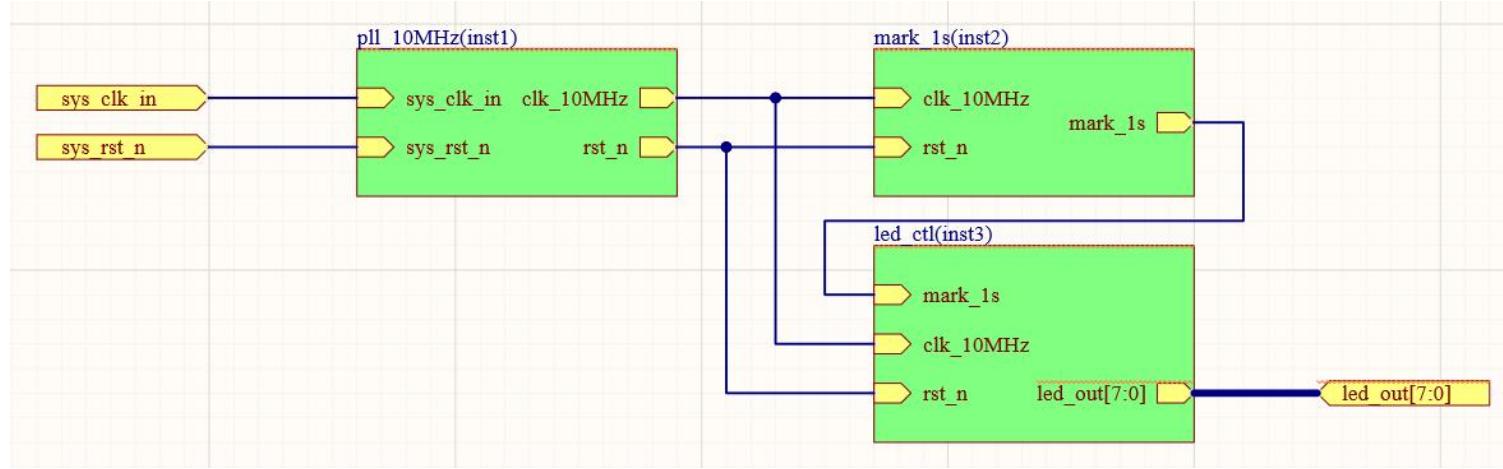
- 使用8颗LED，按照1Hz的频率进行切换点亮
- 效果1：8颗LED按照顺序依次点亮，每次点亮1颗
- 效果2：花式点亮
 - (1) 8颗LED按顺序依次点亮
 - (2) 8颗LED全亮、全灭、全亮
 - (3) 8颗LED从中间的LED开始，每次熄灭2颗
- 效果3：自行设计

■ 实验目的

- 继续熟悉基于Vivado设计工具的设计流程
- 学会层次化程序设计
- 学会连接语句（{}）的使用



Lab3流水灯实验



■ 设计思路及电路结构

- 系统复位（初始化）时输出“0000_0001”信号（点亮1颗LED）
- 使用循环左移位，按照1s周期对“0000_0001”中的“1”进行移位操作
- 设计三个模块：pll模块、秒脉冲产生模块、led控制模块
- 秒脉冲模块对输入的系统时钟进行计数，输出秒脉冲信号；led控制模块实现对led的状态控制
- 模块均采用异步复位

Lab3流水灯实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

输入输出端口表

序号	信号名称	信号类型	输入/输出	信号描述	FPGA引脚	说明
1	sys_clk_in	1bit	输入	系统时钟输入	P17	100MHz输入
2	sys_RST_n	1bit	输入	系统复位输入	P15	低电平复位
3	led_out[7:0]	8bit	输出	LED控制信号输出	K3 M1 L1 K6 J5 H5 H6 K1	高电平点亮

Lab3流水灯实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

模块输入输出端口表——锁相环模块 (pll_10MHz)

序号	信号名称	信号类型	输入/输出	信号描述	说明
1	sys_clk_in	1bit	输入	系统时钟输入	100MHz输入
2	sys_rst_n	1bit	输入	系统复位输入	低电平复位
3	clk_10MHz	1bit	输出	系统主时钟	10MHz时钟输出
4	rst_n	1bit	输出	锁相环时钟锁定输出	高电平锁定

模块输入输出端口表——秒脉冲模块 (mark_1s)

序号	信号名称	信号类型	输入/输出	信号描述	说明
1	clk_10MHz	1bit	输入	系统主时钟	10MHz时钟输入
2	rst_n	1bit	输入	系统主复位	低电平复位
3	mark_1s	1bit	输出	秒脉冲输出	1秒周期，高电平维持1个clk周期

Lab3流水灯实验



模块输入输出端口表——led控制模块 (led_ctl)

序号	信号名称	信号类型	输入/输出	信号描述	说明
1	clk_10MHz	1bit	输入	系统主时钟	10MHz时钟输入
2	rst_n	1bit	输入	系统主复位	低电平复位
3	mark_1s	1bit	输入	秒脉冲输入	1秒周期，高电平维持1个clk周期
4	led_out	8bit	输出	led控制信号	高电平点亮

Lab3流水灯实验



Vivado设计工程, 设计文件列表

序号	文件名称	文件功能	说明
1	lab3_top.v	设计顶层文件	实验3的顶层设计文件
2	pll_10MHz.xci(pll.v)	锁相环模块	inst1, 通过IP管理器生成
3	mark_1s.v	秒脉冲模块	inst2, 自行设计
4	led_ctl.v	LED控制模块	inst3, 自行设计
5	Lab3.xdc	管脚约束文件	用于管脚约束
6	Lab3_sim.v	仿真激励文件	用于功能仿真

Sources x Netlist Device Constraints ? _ □

Design Sources (1)

- lab3_top (lab3_top.v) (3)
 - inst1 : pll_10MHz (pll_10MHz.xci)
 - inst2 : mark_1s (mark_1s.v)
 - inst3 : led_ctl (led_ctl.v)

Module Definition

Module name: lab3_top

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
sys_clk_in	input	<input type="checkbox"/>	0	0
sys_rst_n	input	<input type="checkbox"/>	0	0
led_out	output	<input checked="" type="checkbox"/>	7	0

Lab3流水灯实验



■ 秒脉冲模块

- 设计思路：对输入的10MHz时钟 ($T=100\text{ns}$) 进行计数，定义reg [31:0] counter
counter计数到 (10_000_000-1) 时，mark_1s输出高电平
- 详细设计：变量列表：reg [31:0] counter

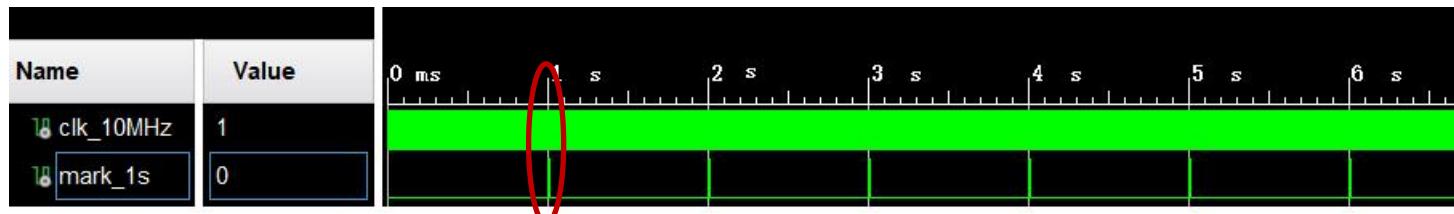
parameter COUNTER_MAX = 32'd10_000_000

功能块：always@(posedge clk_10MHz or negedge rst_n)

若复位信号rst_n有效，则：counter清零，mark_1s置零

否则，若counter == COUNTER_MAX-1，则：counter清零，mark_1s置1

否则，则：counter+1，mark_1s置零



$$\text{计数器清零计数值} = \frac{1\text{s}}{100\text{ns}} = 10 \times 10^6$$





■ led控制模块

- 将led_out初始化为0000_0001
- 每检测到一次秒脉冲 (mark_1s) , 对led_out进行一次循环左移位操作
- 关键语句: `led_out <= {led_out[6:0],led_out[7]}`;
- 解释: (1) led_out为8位向量, 可以使用[]语句进行截取
(2) {a,b}, 大括号为链接语句, 实现a和b信号的链接

流水灯实验效果1

设计思路参考

Lab3流水灯实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

```
module lab3_top(
    input sys_clk_in,
    input sys_rst_n,
    output [7:0] led_out
);

wire clk_10MHz;
wire rst_n;
wire mark_1s;

pll_10MHz inst1
(
    .clk_10MHz(clk_10MHz),           // output clk_10MHz
    .resetn(sys_rst_n),              // input reset
    .rst_n(rst_n),                  // output rst_n
    .sys_clk_in(sys_clk_in)          // input sys_clk_in
);

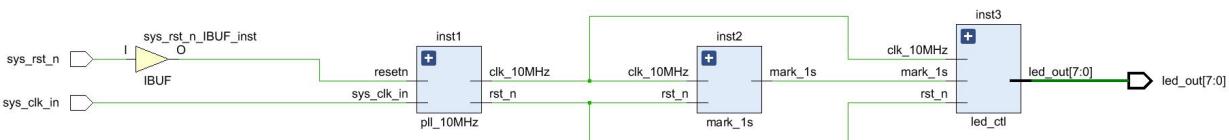
mark_1s inst2
(
    .clk_10MHz(clk_10MHz),
    .rst_n(rst_n),
    .mark_1s(mark_1s)
);

led_ctl inst3
(
    .clk_10MHz(clk_10MHz),
    .rst_n(rst_n),
    .mark_1s(mark_1s),
    .led_out(led_out)
);

endmodule
```

顶层文件
只做模块互联

流水灯实验效果1 参考代码



Lab3流水灯实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

```
module mark_1s(
    input clk_10MHz,
    input rst_n,
    output reg mark_1s
);

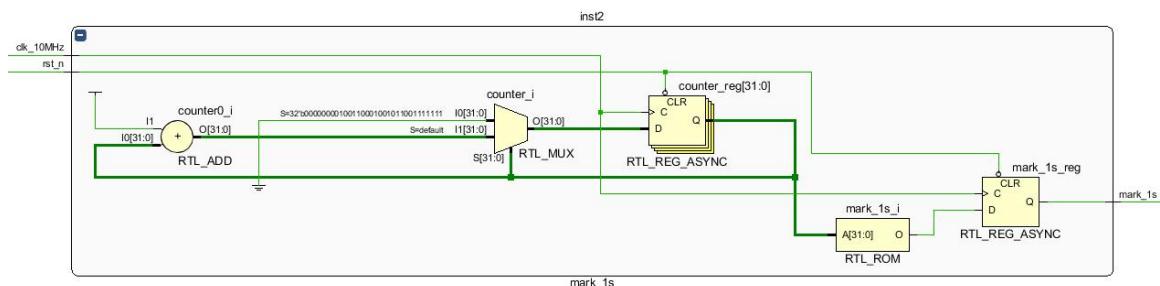
reg [31:0] counter;

parameter COUNTER_MAX = 32'd10_000_000;

always @(posedge clk_10MHz or negedge rst_n) begin
    if(!rst_n) begin
        counter <= 32'd0;
        mark_1s <= 1'b0;
    end
    else if(counter == (COUNTER_MAX -1)) begin
        counter <= 32'd0;
        mark_1s <= 1'b1;
    end
    else begin
        counter <= counter + 1'b1;
        mark_1s <= 1'b0;
    end
end
endmodule
```

```
module led_ctl(
    input clk_10MHz,
    input rst_n,
    input mark_1s,
    output reg [7:0] led_out
);

always @ (posedge clk_10MHz or negedge rst_n ) begin
    if(!rst_n) begin
        led_out <= 8'b0000_0001;
    end
    else if(mark_1s) begin
        led_out <= {led_out[6:0],led_out[7]};
    end
end
endmodule
```

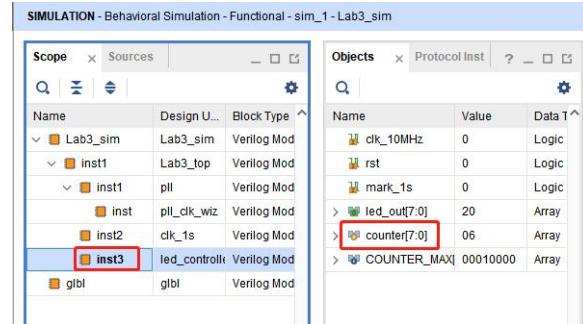
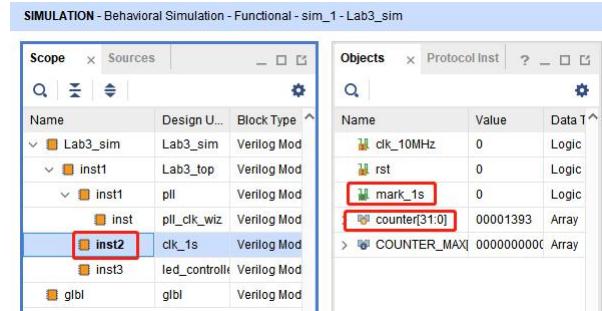
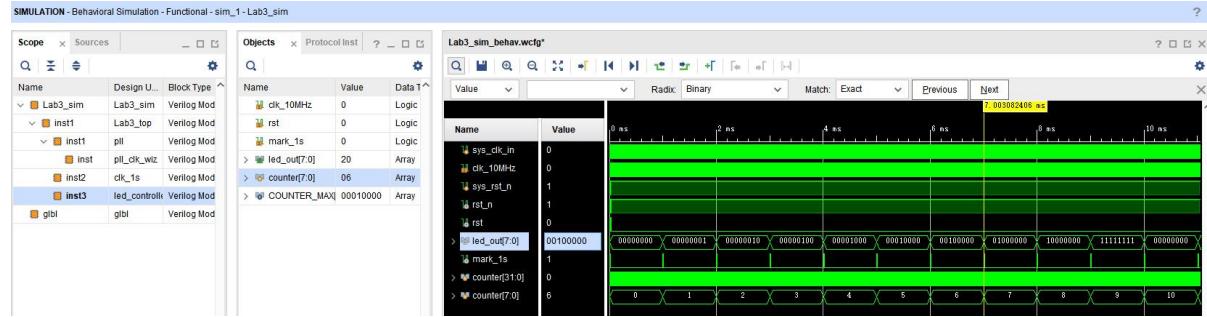


Lab3流水灯实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

功能仿真



```
module Lab3_sim();
    reg sys_clk_in;
    reg sys_rst_n;
    wire [7:0] led_out;

    Lab3_top inst1
    (
        .sys_clk_in(sys_clk_in),
        .sys_rst_n(sys_rst_n),
        .led_out(led_out)
    );

    initial begin
        sys_clk_in = 1'b0;
        sys_rst_n = 1'b0;
        #1000
        sys_rst_n = 1'b1;
    end

    always #5 sys_clk_in = ~sys_clk_in;

endmodule
```

Lab3流水灯实验



仿真小技巧

```
`define SIMULATION 定义仿真标志

module mark_1s(
    input clk_10MHz,
    input rst_n,
    output reg mark_1s
);

reg [31:0] counter;

`ifdef SIMULATION
    parameter COUNTER_MAX = 32'd10_000;
`else
    parameter COUNTER_MAX = 32'd10_000_000;
`endif

always @(posedge clk_10MHz or negedge rst_n) begin
    if(!rst_n) begin
        counter <= 32'd0;
        mark_1s <= 1'b0;
    end
    else if(counter == (COUNTER_MAX -1)) begin
        counter <= 32'd0;
        mark_1s <= 1'b1;
    end
    else begin
        counter <= counter + 1'b1;
        mark_1s <= 1'b0;
    end
end
endmodule
```

如果定义了仿真标志，采用较小的计数值
需要生成烧录代码时，再将仿真标志注释掉

Lab3流水灯实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ led控制模块

- 对秒脉冲信号进行计数，得到秒脉冲计数值
- 使用case语句，根据秒脉冲计数值对led_out进行赋值
- 可实现任意花式的流水灯效果
- 关键语句

```
case(counter_1s)
  8'd0: led_out <= 0000_0000;
  8'd1: led_out <= 0000_0001;
  ...
  ...
  default: led_out <= 0000_0000;
endcase
```

流水灯实验效果2

设计思路参考

课后作业：HDLbits题目



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

- ▶ Getting Started
- ▼ Verilog Language
 - ▶ Basics
 - ▶ Vectors
 - ▶ Modules: Hierarchy
 - ▶ Procedures
 - ▼ More Verilog Features
 - Conditional ternary operator**
 - Reduction operators**
 - Reduction: Even wider gates
 - Combinational for-loop: Vector reversal 2
 - Combinational for-loop: 255-bit population count
 - Generate for-loop: 100-bit binary adder 2
 - Generate for-loop: 100-digit BCD adder
 - ▶ Circuits
 - ▶ Verification: Reading Simulations
 - ▶ Verification: Writing Testbenches
 - ▶ CS450
- ▶ Getting Started
- ▶ Verilog Language
- ▼ Circuits
 - ▶ Combinational Logic
 - ▼ Sequential Logic
 - ▶ Latches and Flip-Flops
 - ▼ Counters
 - Four-bit binary counter
 - Decade counter
 - Decade counter again
 - Slow decade counter
 - Counter 1-12
 - Counter 1000
 - 4-digit decimal counter**
 - 12-hour clock
 - ▶ Shift Registers
 - ▶ More Circuits
 - ▶ Finite State Machines
 - ▶ Building Larger Circuits
 - ▶ Verification: Reading Simulations
 - ▶ Verification: Writing Testbenches
 - ▶ CS450

下节预告



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Lab4实验内容

Lab4数码管实验1



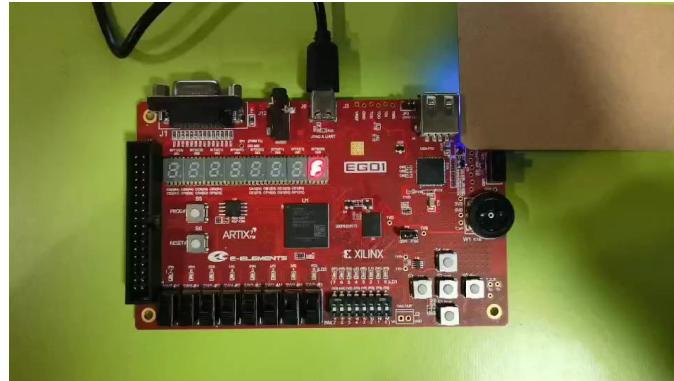
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 实验内容

- 学会数码管的控制原理
- 完成数码管的使用控制
- 效果1：单颗数码管控制
 - (1) 数码管字库建立
 - (2) 数码管驱动控制
- 效果2：两颗数码管控制
 - (1) 数码管驱动程序编写
 - (2) 数码管动态显示

■ 实验目的

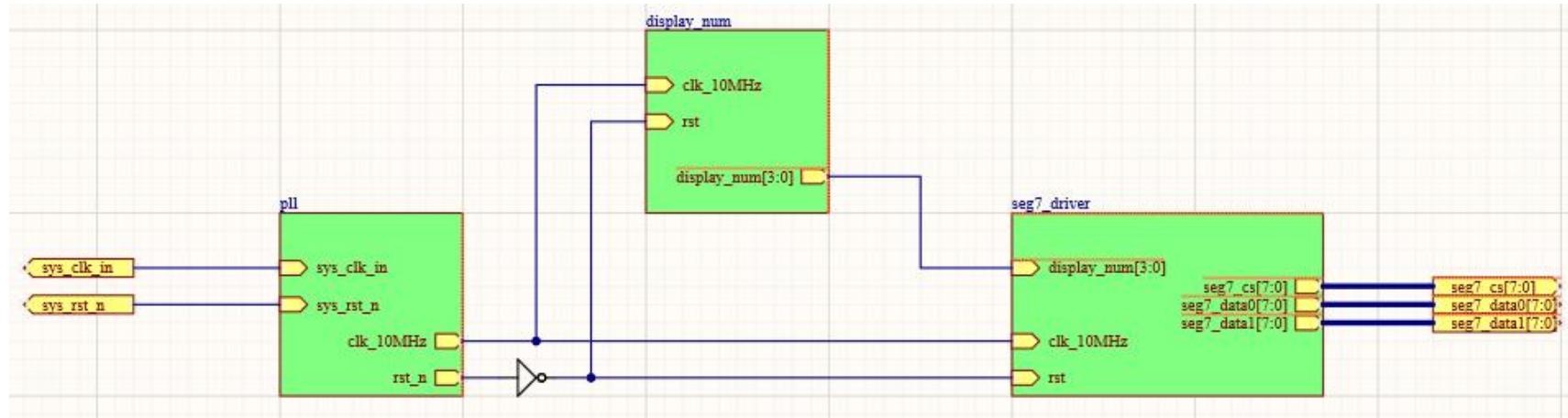
- 进一步理解层次化程序设计
- 熟练使用时钟管理IP核



Lab4数码管实验1



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



■ 系统框图

- 电路包括三个模块：pll模块实现时钟管理，display_num模块产生需要显示的数据，seg7_driver模块完成数码管驱动控制
- pll模块：通过IP核实现，100MHz时钟输入，全局复位引脚输入，生成10MHz信号，频率锁定信号用于后续模块复位
- display模块：10MHz时钟输入，频率锁定信号作为复位信号，输出需要显示的数据（每位数码管数据用4位表示）
- seg7_driver模块：10MHz时钟输入，频率锁定信号作为复位信号，通过seg7_cs,seg7_data控制数码管点亮



THANKS



國科大杭州高等研究院
Hangzhou Institute for Advanced Study, UCAS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第六讲 (6.1 Verilog语法3)

2025年4月8日



- 1. 模块例化参数化设计**
- 2. 仿真语法概述**
- 3. Verilog代码规范**
- 4. Verilog国际标准概述**



1. 模块例化参数化设计

Verilog语法3



■参数化设计

- 模块内部采用参数化设计
- 顶层对模块采用参数化例化
- 参数放入统一的define文件
- 优点：便于程序的维护与复用

```
``include "parameter_define.v"
```

```
`define COUNTER_100MS 32'd5_000_000
`define BPS_CNT_MAX 16'd5208
```

全局参数可以放入define文件

```
module ADC_Driver #(
    parameter DATA_WIDTH = 16,
    parameter ADDR_WIDTH = 8
) (
    input [DATA_WIDTH:0] data,
    output [ADDR_WIDTH:0] addr
);
    端口位宽的参数化设计
    //coding.....
endmodule
```

```
module data_gen
#(
    parameter COUNTER_100MS = 32'd5_000_000,
    parameter BPS_CNT_MAX = 16'd10471
)
(
    input clk,
    input rst,
    input full,
    output reg wr_en,
    output reg [7:0] gen_data
);
```

```
data_gen #(
    .COUNTER_100MS(`COUNTER_100MS),
    .BPS_CNT_MAX(`BPS_CNT_MAX)
)
inst2_data_gen
(
    .clk(clk_50MHz),
    .rst(rst),
    .full(fifo_full),
    .wr_en(fifo_wr_en),
    .gen_data(gen_data)
);
```

■带参数值模块例化

- `define: 用于跨文件使用，参数使用时需要加“`”
- parameter: 建议放到端口中定义，便于参数化设计
- localparam: 用于模块内部的参数化设计，不能被外部模块修改
- 示例：实现端口位宽参数化设计

```
parameter COUNTER_MAX = 10_000_000;
localparam COUNTER_MAX = 10_000_000;
```



2. 仿真语法概述

Verilog语法3



■ 使用Testbench激励文件

- TestBench也是由HDL语言代码编写，它实例化了需要仿真设计，生成设计所需要的激励信号，监测设计输出结果并检查功能的正确性；
- 简单的TestBench可以仅仅将激励顺序地加载到设计的输入信号上；
- 复杂的TestBench可能会包含子程序调用、从外部文件读取激励信号、条件化激励和其它更多复杂的结构。

■ Testbench编写注意事项

- 在Verilog TestBench中总是使用timescale规定时间，如`timescale 1ns/1ps
- 注意Testbench编写的大部分不可综合，不能与可综合语法混淆

```
1 `timescale 1ns / 1ps // 单位时间/时间精度
2 module uart_tb();
3
4     //定义信号
5     reg Clk;
6     reg Rst_n;
7     wire uart_tx;
8
9     //实例化
10    uart_tx uart_tx_tb(
11        .clk(Clk),           // 100MHz 系统时钟
12        .reset(Rst_n),      // 复位信号
13        .tx(uart_tx)        // 输出端口
14    );
15
16    initial Clk = 1;       //初始时钟
17    always #5 Clk = ~Clk; //每隔5ns反转一次，即产生100MHz时钟
18    initial begin
19        Rst_n = 0;
20        #200;
21        Rst_n = 1;
22    end
23 endmodule
```

Verilog语法3



■ 时钟生成

- 生成时钟的方式有很多种
- 利用取反方法产生时钟时，一定要给 clk 寄存器赋初值
- 如果延时参数为浮点数，该参数不要声明为 parameter 类型，可以使用real型
- 如果延时参数为浮点数，timescale 的精度也需要提高，单位和精度不能相同，否则小数部分的时间延迟赋值将不起作用

```
parameter CLOCK_PERIOD = 10; //定义仿真时钟周期  
  
initial clk = 0 ;           时钟产生方式1 //定义仿真时钟初始值  
always #(CLOCK_PERIOD/2) clk = ~clk; //生成仿真时钟
```

```
parameter CLOCK_PERIOD = 10; //定义仿真时钟周期  
  
initial begin  
    clk = 0 ;           时钟产生方式2 //定义仿真时钟初始值  
    forever begin  
        #(CLOCK_PERIOD/2) clk = ~clk; //生成仿真时钟  
    end  
end
```

```
parameter CLOCK_PERIOD_HIGH = 7; //定义仿真时钟高电平时间  
parameter CLOCK_PERIOD_LOW = 3; //定义仿真时钟低电平时间  
  
always begin               带占空比时钟产生方式  
    clk = 1;  
    #(CLOCK_PERIOD_HIGH)  
    clk = 0;  
    #(CLOCK_PERIOD_LOW)  
end
```



■ 复杂激励的产生

- 复杂激励的情况下，通过简单延时产生波形的方式效率较低
- 可以通过加载文本的方式导入激励波形
- 激励波形可通过第三方软件（Matlab/LabVIEW等）生成

■ 文件操作

- 文件打开/关闭：\$fopen, \$fclose, \$ferror
- 文件写入：\$fdisplay, \$fwrite, \$fstrobe, \$fmonitor
- 字符串写入：\$sformat, \$swrite
- 文件读取：\$fgetc, \$fgets, \$fscanf, \$fread
- 文件定位：\$fseek, \$ftell, \$feof, \$rewind
- 存储器加载：\$readmemh, \$readmemb

```
integer file;          打开文件
file = $fopen("input.bin", "r");
```

```
reg [7:0] data;
while (!feof(file)) begin           //判断是否读到文件末尾
    @(negedge clk);                //等待clk的下降沿
    $fread(data, file);           //按照二进制方式从文件中读取跟1个跟data类型一致的数据，文件指针+1
end
```

```
$fclose(file);                  关闭文件
```



■ 仿真自动校验

- 简单设计：可以通过输入、输出信号的波形来确定设计是否正确
- 复杂设计：加入自校验模块，会大大增加仿真的效率，也可以通过文件记录的方式将仿真结果写入文件

■ 仿真控制

- \$finish(type)：结束仿真，参数 type 可选择退出仿真时是否打印信息；
- \$stop(type)：暂停仿真，参数 type 可选择暂停仿真时是否打印信息
- 参数 type含义：0-不打印；1-打印仿真时间及该语句所在行；2-打印仿真时间、位置、存储器和 CPU 时间的使用情况

```
integer file;
file = $fopen("output.bin", "r");      //打开判断文件

reg [7:0] data;
reg [15:0] err_cnt;

while (!$feof(file)) begin
    @(negedge clk);
    $fread(data, file);           //判断是否读到文件末尾
                                    //等待clk的下降沿
                                    //按照二进制方式从文件中读取跟data类型一致的数据，文件指针+1
end

always @(posedge clk) begin
    #1 ;
    if (data_out != data) begin   //对仿真数据进行自动比对
        err_cnt = err_cnt + 1'b1; //如果与仿真数据不一致则错误计数+1
        $display("Find error !!!"); //打印错误信息
    end
end

fclose(file);
```

Verilog语法3



■ 其他方便仿真的功能

- **\$display:** 类似于C语言的printf
- **\$monitor:** 提供了监控和输出参数列表中的表达式或变量值的功能
- **\$time:** 获得一个64位的integer型变量，其表示调用该系统任务时的仿真时间
- **\$random:** 产生随机数的系统函数，每次调用该函数将返回一个32位的带符号的整形随机数

```
$display("the code is running...");  
the code is running...
```

```
$display("data0= %b;data1= %o ; data2= %d; data3= %h",4, 6 ,12,13);  
data0= 100;data1= 6 ; data2= 12; data3= d
```

```
initial begin  
    $monitor($time, ": a=%3d, b=%3d", a, b);  
end
```

```
VSIM 4> run -all  
#                                     0: a= 2, b= 4  
#                                     5: a= 6, b= 4  
#                                     10: a= 6, b= 5
```

\$random % b 得到-b+1至b-1之间的随机数

{\$random} % b 得到0至b-1之间的随机数



3. Verilog代码规范

Verilog语法3



■ 代码规范

- 初学者编写 Verilog 代码，容易按照 C 语言的思维和风格去设计，造成了很多不规范的共性问题
- 代码规范与代码风格不同，**编码风格是一种建议，代码规范是在一定程度上必须要遵从的规则**

```
reg [31:0] data = 32'b0 ;\n\nalways @ (posedge clk or negedge clk) begin\n    |   a <= b ;\nend\n\nalways @ (posedge clk) begin          X\n    |   a <= b ;\nend\nalways @ (negedge clk) begin\n    |   a <= d ;\nend
```

■ 重要的代码规范

- 符合Verilog语法规则
- 尽可能采用同步时序，异步设计要重视跨时钟域信号
- 禁止在多个 always 块中为同一个变量赋值：电路无法实现**
- 禁止在多个 always 块中分别使用同一时钟的上升沿和下降沿：会引入相对复杂的时钟质量和时序约束的问题**
- 禁止在一个 always 块中同时将时钟的双边沿作为触发条件：如需要请使用双边沿硬件原语**
- 避免变量声明时对变量赋初值：仿真时变量会有期望的初值，但综合后电路的初始值是不确定的，建议通过复位赋初值**
- 避免直接将时钟信号与普通变量信号做逻辑操作：容易产生毛刺**
- 避免直接使用乘法 *、除法 /、求余数 % 等操作：结构和时序不易控制，优先通过IP方式实现**
- 避免组合逻辑的条件语句中判断条件不完整：避免产生锁存器，易产生毛刺**
- 避免组合逻辑的 always 块的敏感信号为列全：影响行为级仿真**
- 避免使用门控时钟及行波时钟：时序不易控制**

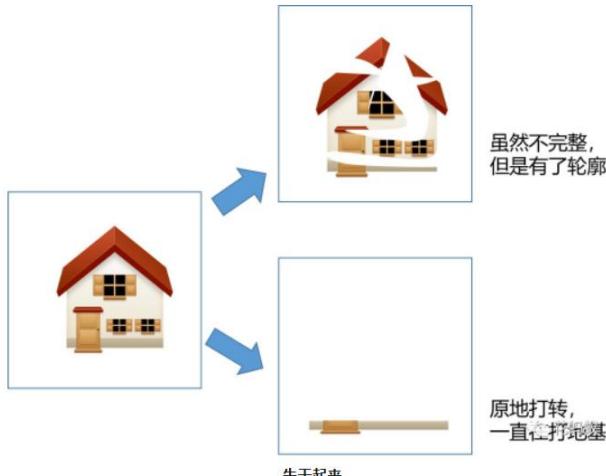


4. Verilog国际标准概述

Verilog语法3



- ◆ 第一次成为国际标准: IEEE Std 1364-1995
- ◆ 第一次修订版本: IEEE Std 1364-2001
- ◆ 第二次修订版本: IEEE Std 1364-2005
- ◆ IEEE Std 1364-2005全文共590页
- ◆ 好消息是: 可综合的语法占比不超过30%, 其余主要用于仿真及测试



1364™

IEEE
3 Park Avenue
New York, NY 10016-5997, USA
7 April 2006

IEEE Std 1364™-2005
(Revision of IEEE Std 1364-2001)

IEEE Standard for Verilog® Hardware Description Language

IEEE Computer Society

Sponsored by the
Design Automation Standards Committee

Verilog语法3



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

- ◆ 第一章，概述
- ◆ 第二章，引用参考
- ◆ 第三章，词汇惯例：使用词汇的一些说明
- ◆ 第四章，数据类型：wire、reg、vector、强度、隐式声明、整形、数组等

Contents

1.	Overview.....	1
1.1	Scope.....	1
1.2	Conventions used in this standard	1
1.3	Syntactic description	2
1.4	Use of color in this standard	3
1.5	Contents of this standard	3
1.6	Deprecated clauses	5
1.7	Header file listings	5
1.8	Examples	5
1.9	Prerequisites	5
2.	Normative references.....	6
3.	Lexical conventions.....	8
3.1	Lexical tokens	8
3.2	White space	8
3.3	Comments	8
3.4	Operators	8
3.5	Numbers	9
3.5.1	Integer constants	10
3.5.2	Real constants	12
3.5.3	Conversion	12
3.6	Strings	12
3.6.1	String variable declaration	13
3.6.2	String manipulation	13
3.6.3	Special characters in strings	13
3.7	Identifiers, keywords, and system names	14
3.7.1	Escaped identifiers	14
3.7.2	Keywords	15
3.7.3	System tasks and functions	15
3.7.4	Compiler directives	15
3.8	Attributes	16
3.8.1	Examples	16
3.8.2	Syntax	18
4.	Data types	21
4.1	Value set	21
4.2	Nets and variables	21
4.2.1	Net declarations	21
4.2.2	Variable declarations	23
4.3	Vectors	24
4.3.1	Specifying vectors	24
4.3.2	Vector net accessibility	24
4.4	Strengths	25
4.4.1	Charge strength	25
4.4.2	Drive strength	25
4.5	Implicit declarations	25
4.6	Net types	26
4.6.1	Wire and tri nets	26
4.6.2	Wired nets	27
4.6.3	Trireg net	28

Verilog语法3



- ◆ 第五章, 表达式: 操作数、运算符
- ◆ 第六章, 赋值: 连续赋值 (continuous) 、过程赋值 (procedural)

4.6.4	Tri0 and tri1 nets.....	31
4.6.5	Unresolved nets.....	31
4.6.6	Supply nets.....	32
4.7	Regs	32
4.8	Integers, reals, times, and realtimes.....	32
4.8.1	Operators and real numbers.....	33
4.8.2	Conversion	33
4.9	Arrays.....	34
4.9.1	Net arrays	34
4.9.2	reg and variable arrays	34
4.9.3	Memories	35
4.10	Parameters.....	35
4.10.1	Module parameters	36
4.10.2	Local parameters (localparam)	37
4.10.3	Specify parameters.....	38
4.11	Name spaces	39
5.	Expressions	41
5.1	Operators	41
5.1.1	Operators with real operands	42
5.1.2	Operator precedence	43
5.1.3	Using integer numbers in expressions	44
5.1.4	Expression evaluation order	45
5.1.5	Arithmetic operators	45
5.1.6	Arithmetic expressions with regs and integers	47
5.1.7	Relational operators	48
5.1.8	Equality operators	49
5.1.9	Logical operators	49
5.1.10	Bitwise operators	50
5.1.11	Reduction operators	51
5.1.12	Shift operators	53
5.1.13	Conditional operator	53
5.1.14	Concatenations	54
5.2	Operands	55
5.2.1	Vector bit-select and part-select addressing	56
5.2.2	Array and memory addressing	57
5.2.3	Strings	58
5.3	Minimum, typical, and maximum delay expressions	61
5.4	Expression bit lengths	62
5.4.1	Rules for expression bit lengths	62
5.4.2	Example of expression bit-length problem	63
5.4.3	Example of self-determined expressions	64
5.5	Signed expressions	64
5.5.1	Rules for expression types	65
5.5.2	Steps for evaluating an expression	65
5.5.3	Steps for evaluating an assignment	66
5.5.4	Handling X and Z in signed expressions	66
5.6	Assignments and truncation	66
6.	Assignments	68
6.1	Continuous assignments	68
6.1.1	The net declaration assignment	69
6.1.2	The continuous assignment statement	69
6.1.3	Delays	71

Verilog语法3



◆ 第七章，门级和开关级建模：门级及开关级原语、硬件描述方法

◆ 第八章，用户定义原语：自定义原语级，与门级及开关级原语一样用于电路建模

6.1.4	Strength	71
6.2	Procedural assignments.....	72
6.2.1	Variable declaration assignment	72
6.2.2	Variable declaration syntax	73
7.	Gate- and switch-level modeling	74
7.1	Gate and switch declaration syntax	74
7.1.1	The gate type specification	76
7.1.2	The drive strength specification	76
7.1.3	The delay specification	77
7.1.4	The primitive instance identifier	77
7.1.5	The range specification	77
7.1.6	Primitive instance connection list	78
7.2	and, nand, nor, or, xor, and xnor gates	80
7.3	buf and not gates	81
7.4	bufif1, bufif0, notif1, and notif0 gates	82
7.5	MOS switches	83
7.6	Bidirectional pass switches	84
7.7	CMOS switches	85
7.8	pullup and pulldown sources	86
7.9	Logic strength modeling	86
7.10	Strengths and values of combined signals	88
7.10.1	Combined signals of unambiguous strength	88
7.10.2	Ambiguous strengths: sources and combinations	89
7.10.3	Ambiguous strength signals and unambiguous signals	94
7.10.4	Wired logic net types	98
7.11	Strength reduction by nonresistive devices	100
7.12	Strength reduction by resistive devices	100
7.13	Strengths of net types	100
7.13.1	tri0 and tri1 net strengths	100
7.13.2	trireg strength	100
7.13.3	supply0 and supply1 net strengths	101
7.14	Gate and net delays	101
7.14.1	min:typ:max delays	102
7.14.2	trireg net charge decay	103
8.	User-defined primitives (UDPs)	105
8.1	UDP definition	105
8.1.1	UDP header	107
8.1.2	UDP port declarations	107
8.1.3	Sequential UDP initial statement	107
8.1.4	UDP state table	107
8.1.5	Z values in UDP	108
8.1.6	Summary of symbols	108
8.2	Combinational UDPs	109
8.3	Level-sensitive sequential UDPs	110
8.4	Edge-sensitive sequential UDPs	110
8.5	Sequential UDP initialization	111
8.6	UDP instances	113
8.7	Mixing level-sensitive and edge-sensitive descriptions	114
8.8	Level-sensitive dominance	115
9.	Behavioral modeling	116
9.1	Behavioral model overview	116

Verilog语法3



- ◆ 第九章，行为级建模：if else、case、循环等
- ◆ 第十章，任务及函数：便于大型程序设计
- ◆ 第十一章，调度语义：通过调度串行事件模拟芯片的真实行为

9.2	Procedural assignments.....	117
9.2.1	Blocking procedural assignments.....	117
9.2.2	The nonblocking procedural assignment.....	118
9.3	Procedural continuous assignments.....	122
9.3.1	The assign and deassign procedural statements.....	123
9.3.2	The force and release procedural statements.....	124
9.4	Conditional statement.....	125
9.4.1	If-else-if construct.....	126
9.5	Case statement.....	127
9.5.1	Case statement with do-not-cares.....	128
9.5.2	Constant expression in case statement.....	129
9.6	Looping statements.....	130
9.7	Procedural timing controls.....	131
9.7.1	Delay control.....	132
9.7.2	Event control.....	132
9.7.3	Named events.....	133
9.7.4	Event or operator.....	134
9.7.5	Implicit event expression list.....	134
9.7.6	Level-sensitive event control.....	136
9.7.7	Intra-assignment timing controls.....	136
9.8	Block statements.....	139
9.8.1	Sequential blocks.....	140
9.8.2	Parallel blocks.....	141
9.8.3	Block names.....	141
9.8.4	Start and finish times.....	142
9.9	Structured procedures.....	143
9.9.1	Initial construct.....	143
9.9.2	Always construct.....	144
10.	Tasks and functions.....	145
10.1	Distinctions between tasks and functions.....	145
10.2	Tasks and task enabling.....	145
10.2.1	Task declarations.....	146
10.2.2	Task enabling and argument passing.....	147
10.2.3	Task memory usage and concurrent activation.....	149
10.3	Disabling of named blocks and tasks.....	150
10.4	Functions and function calling.....	152
10.4.1	Function declarations.....	152
10.4.2	Returning a value from a function.....	154
10.4.3	Calling a function.....	155
10.4.4	Function rules.....	155
10.4.5	Use of constant functions.....	156
11.	Scheduling semantics.....	158
11.1	Execution of a model.....	158
11.2	Event simulation.....	158
11.3	The stratified event queue.....	158
11.4	Verilog simulation reference model.....	159
11.4.1	Determinism.....	160
11.4.2	Nondeterminism.....	160
11.5	Race conditions.....	160
11.6	Scheduling implication of assignments.....	161
11.6.1	Continuous assignment.....	161
11.6.2	Procedural continuous assignment.....	161



◆ 第十二章，层次结构

◆ 第十三章，设计的配置：库、配置

11.6.3	Blocking assignment	161
11.6.4	Nonblocking assignment	161
11.6.5	Switch (transistor) processing	161
11.6.6	Port connections	162
11.6.7	Functions and tasks	162
12.	Hierarchical structures	163
12.1	Modules	163
12.1.1	Top-level modules	165
12.1.2	Module instantiation	165
12.2	Overriding module parameter values	167
12.2.1	defparam statement	168
12.2.2	Module instance parameter value assignment	170
12.2.3	Parameter dependence	173
12.3	Ports	173
12.3.1	Port definition	173
12.3.2	List of ports	174
12.3.3	Port declarations	174
12.3.4	List of port declarations	176
12.3.5	Connecting module instance ports by ordered list	176
12.3.6	Connecting module instance ports by name	177
12.3.7	Real numbers in port connections	178
12.3.8	Connecting dissimilar ports	178
12.3.9	Port connection rules	179
12.3.10	Net types resulting from dissimilar port connections	179
12.3.11	Connecting signed values via ports	181
12.4	Generate constructs	181
12.4.1	Loop generate constructs	183
12.4.2	Conditional generate constructs	186
12.4.3	External names for unnamed generate blocks	190
12.5	Hierarchical names	191
12.6	Upwards name referencing	193
12.7	Scope rules	195
12.8	Elaboration	197
12.8.1	Order of elaboration	197
12.8.2	Early resolution of hierarchical names	197
13.	Configuring the contents of a design	199
13.1	Introduction	199
13.1.1	Library notation	199
13.1.2	Basic configuration elements	200
13.2	Libraries	200
13.2.1	Specifying libraries—the library map file	200
13.2.2	Using multiple library map files	202
13.2.3	Mapping source files to libraries	202
13.3	Configurations	202
13.3.1	Basic configuration syntax	202
13.3.2	Hierarchical configurations	205
13.4	Using libraries and configs	205
13.4.1	Precompiling in a single-pass use model	205
13.4.2	Elaboration-time compiling in a single-pass use model	206
13.4.3	Precompiling using a separate compilation tool	206
13.4.4	Command line considerations	206
13.5	Configuration examples	206

Verilog语法3



- ◆ 第十四章，特殊块：描述从源点到终点的路径延时
- ◆ 第十五章，时序检查

13.5.1	Default configuration from library map file	207
13.5.2	Using default clause	207
13.5.3	Using cell clause	207
13.5.4	Using instance clause	208
13.5.5	Using hierarchical config	208
13.6	Displaying library binding information	208
13.7	Library mapping examples	209
13.7.1	Using the command line to control library searching	209
13.7.2	File path specification examples	209
13.7.3	Resolving multiple path specifications	209
14.	Specify blocks	211
14.1	Specify block declaration	211
14.2	Module path declarations	212
14.2.1	Module path restrictions	212
14.2.2	Simple module paths	213
14.2.3	Edge-sensitive paths	214
14.2.4	State-dependent paths	215
14.2.5	Full connection and parallel connection paths	219
14.2.6	Declaring multiple module paths in a single statement	220
14.2.7	Module path polarity	220
14.3	Assigning delays to module paths	222
14.3.1	Specifying transition delays on module paths	222
14.3.2	Specifying x transition delays	224
14.3.3	Delay selection	225
14.4	Mixing module path delays and distributed delays	225
14.5	Driving wired logic	226
14.6	Detailed control of pulse filtering behavior	228
14.6.1	Specify block control of pulse limit values	229
14.6.2	Global control of pulse limit values	230
14.6.3	SDF annotation of pulse limit values	230
14.6.4	Detailed pulse control capabilities	230
15.	Timing checks	237
15.1	Overview	237
15.2	Timing checks using a stability window	240
15.2.1	\$setup	241
15.2.2	\$hold	242
15.2.3	\$setuphold	243
15.2.4	\$removal	245
15.2.5	\$recovery	246
15.2.6	\$crem	247
15.3	Timing checks for clock and control signals	248
15.3.1	\$skew	249
15.3.2	\$timeskew	250
15.3.3	\$fullskew	252
15.3.4	\$width	255
15.3.5	\$period	256
15.3.6	\$nochange	257
15.4	Edge-control specifiers	258
15.5	Notifiers: user-defined responses to timing violations	259
15.5.1	Requirements for accurate simulation	261
15.5.2	Conditions in negative timing checks	263
15.5.3	Notifiers in negative timing checks	264

Verilog语法3



◆ 第十六章，使用标准延迟格式的反向注释

用作各种工具之间的文本类型的时序信息交换媒介

◆ 第十七章，系统任务和函数

15.5.4	Option behavior	264
15.6	Enabling timing checks with conditioned events	265
15.7	Vector signals in timing checks	266
15.8	Negative timing checks	266
16.	Backannotation using the standard delay format (SDF)	269
16.1	The SDF annotator	269
16.2	Mapping of SDF constructs to Verilog	269
16.2.1	Mapping of SDF delay constructs to Verilog declarations	269
16.2.2	Mapping of SDF timing check constructs to Verilog	271
16.2.3	SDF annotation of specparams	272
16.2.4	SDF annotation of interconnect delays	273
16.3	Multiple annotations	274
16.4	Multiple SDF files	275
16.5	Pulse limit annotation	275
16.6	SDF to Verilog delay value mapping	276
17.	System tasks and functions	277
17.1	Display system tasks	278
17.1.1	The display and write tasks	278
17.1.2	Strobed monitoring	285
17.1.3	Continuous monitoring	286
17.2	File input-output system tasks and functions	286
17.2.1	Opening and closing files	287
17.2.2	File output system tasks	288
17.2.3	Formatting data to a string	289
17.2.4	Reading data from a file	290
17.2.5	File positioning	294
17.2.6	Flushing output	295
17.2.7	I/O error status	295
17.2.8	Detecting EOF	295
17.2.9	Loading memory data from a file	296
17.2.10	Loading timing data from an SDF file	297
17.3	Timescale system tasks	298
17.3.1	\$printtimescale	299
17.3.2	\$timeformat	300
17.4	Simulation control system tasks	302
17.4.1	\$finish	302
17.4.2	\$stop	302
17.5	Programmable logic array (PLA) modeling system tasks	303
17.5.1	Array types	303
17.5.2	Array logic types	304
17.5.3	Logic array personality declaration and loading	304
17.5.4	Logic array personality formats	304
17.6	Stochastic analysis tasks	307
17.6.1	\$q_initialize	307
17.6.2	\$q_add	307
17.6.3	\$q_remove	307
17.6.4	\$q_fill	308
17.6.5	\$q_exam	308
17.6.6	Status codes	308
17.7	Simulation time system functions	309
17.7.1	\$time	309
17.7.2	\$stime	309

Verilog语法3



◆ 第十八章，值更改转储文件 (VCD)

VCD文件是一种标准格式的波形记录文件，只记录发生变化的波形

◆ 第十九章，编译器指令

17.7.3	\$realtime	310
17.8	Conversion functions	310
17.9	Probabilistic distribution functions	311
17.9.1	\$random function	311
17.9.2	\$dist_ functions	312
17.9.3	Algorithm for probabilistic distribution functions	313
17.10	Command line input	320
17.10.1	\$test\$plusargs (string)	320
17.10.2	\$value\$plusargs (user_string, variable)	321
17.11	Math functions	323
17.11.1	Integer math functions	323
17.11.2	Real math functions	323
18.	Value change dump (VCD) files	325
18.1	Creating four-state VCD file	325
18.1.1	Specifying name of dump file (\$dumpfile)	325
18.1.2	Specifying variables to be dumped (\$dumpvars)	326
18.1.3	Stopping and resuming the dump (\$dumpoff/\$dumpon)	327
18.1.4	Generating a checkpoint (\$dumpall)	328
18.1.5	Limiting size of dump file (\$dumplimit)	328
18.1.6	Reading dump file during simulation (\$dumpflush)	328
18.2	Format of four-state VCD file	329
18.2.1	Syntax of four-state VCD file	330
18.2.2	Formats of variable values	331
18.2.3	Description of keyword commands	332
18.2.4	Four-state VCD file format example	337
18.3	Creating extended VCD file	338
18.3.1	Specifying dump file name and ports to be dumped (\$dumpports)	338
18.3.2	Stopping and resuming the dump (\$dumpportsoff/\$dumpports)	339
18.3.3	Generating a checkpoint (\$dumpportall)	340
18.3.4	Limiting size of dump file (\$dumpportslimit)	340
18.3.5	Reading dump file during simulation (\$dumpportflush)	341
18.3.6	Description of keyword commands	341
18.3.7	General rules for extended VCD system tasks	341
18.4	Format of extended VCD file	342
18.4.1	Syntax of extended VCD file	342
18.4.2	Extended VCD node information	344
18.4.3	Value changes	346
18.4.4	Extended VCD file format example	347
19.	Compiler directives	349
19.1	'celldesign and 'endcelldesign	349
19.2	'default_notype	349
19.3	'define and 'undef	350
19.3.1	'define	350
19.3.2	'undef	352
19.4	'ifdef, 'else, 'elsif, 'endif, 'ifndef	352
19.5	'include	356
19.6	'restart	356
19.7	'line	357
19.8	'timescale	358
19.9	'unconnected_drive and 'nunconnected_drive	360
19.10	'pragma	360
19.10.1	Standard pragmas	361

Verilog语法3



◆ 第二十章，编程语言接口

是一种Verilog代码调用C/C++函数的机制

◆ 第二十一章~第二十五章，取消

◆ 第二十六章，Verilog过程接口

使数字电路的行为级描述代码直接调用C语言的函数

◆ 第二十七章，Verilog过程接口定义

◆ 第二十八章，被保护的封装

为源语言处理器提供加密算法、密钥、包络属性和文本设计数据的规范

19.11 `begin_keywords, `end_keywords.....	361
20. Programming language interface (PLI) overview	366
20.1 PLI purpose and history	366
20.2 User-defined system task/function names	367
20.3 User-defined system task/function types	367
20.4 Overriding built-in system task/function names	367
20.5 User-supplied PLI applications	367
20.6 PLI mechanism	368
20.7 User-defined system task/function arguments	368
20.8 PLI include files	368
21. PLI TF and ACC interface mechanism (deprecated).....	369
22. Using ACC routines (deprecated).....	370
23. ACC routine definitions (deprecated).....	371
24. Using TF routines (deprecated)	372
25. TF routine definitions (deprecated)	373
26. Using Verilog procedural interface (VPI) routines.....	374
26.1 VPI system tasks and functions	374
26.1.1 `siftef VPI application routine	374
26.1.2 `completf VPI application routine	374
26.1.3 `calltf VPI application routine	375
26.1.4 Arguments to siftef, completf, and calltf application routines	375
26.2 VPI mechanism	375
26.2.1 VPI callbacks	375
26.2.2 VPI access to Verilog HDL objects and simulation objects	376
26.2.3 Error handling	376
26.2.4 Function availability	376
26.2.5 Traversing expressions	377
26.3 VPI object classifications	377
26.3.1 Accessing object relationships and properties	378
26.3.2 Object type properties	379
26.3.3 Object file and line properties	380
26.3.4 Delays and values	380
26.3.5 Object protection properties	381
26.4 List of VPI routines by functional category	381
26.5 Key to data model diagrams	383
26.5.1 Diagram key for objects and classes	384
26.5.2 Diagram key for accessing properties	384
26.5.3 Diagram key for traversing relationships	385
26.6 Object data model diagrams	386
26.6.1 Module	387
26.6.2 Instance arrays	388
26.6.3 Scope	389
26.6.4 IO declaration	389
26.6.5 Ports	390
26.6.6 Nets and net arrays	391
26.6.7 Regs and reg arrays	393
26.6.8 Variables	395



THANKS



國科大杭州高等研究院
Hangzhou Institute for Advanced Study, UCAS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS



FPGA电路软硬件设计 第六讲 (6.2 Lab4数码管实验1)

2025年4月8日

Lab4数码管实验1



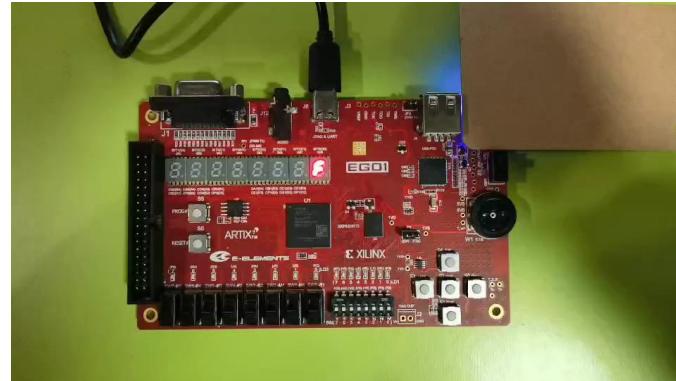
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 实验内容

- 学会数码管的控制原理
- 完成数码管的使用控制
- 效果1：单颗数码管控制
 - (1) 数码管字库建立
 - (2) 数码管驱动控制
- 效果2：两颗数码管控制
 - (1) 数码管驱动程序编写
 - (2) 数码管动态显示

■ 实验目的

- 进一步理解层次化程序设计
- 熟练使用时钟管理IP核



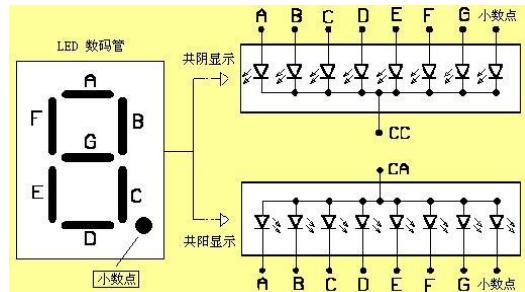
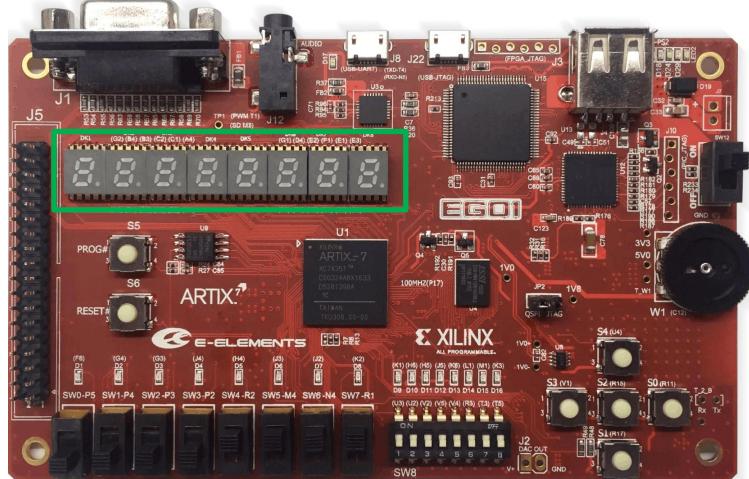
Lab4数码管实验1



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 数码管介绍

- 数码管是一种半导体发光器件，其基本单元是发光二极管
- 能显示4位的称为4位数码管
- 数码管按照发光二极管的单元连接方式分为共阳极和共阴极
- **共阳数码管**是指将所有发光二极管的阳极连接到一起形成**公共阳极**，使用时将**公共阳极COM**接到电源端，当某一字段发光二极管的阴极为**低电平时**，**相应字段就点亮**
- **共阴数码管**是指将所有发光二极管的阴极连接到一起形成**公共阴极**，使用时将**公共阴极COM**接到**地线GND**上，当某一字段发光二极管的阳极为**高电平时**，**相应字段就点亮**
- 字段引脚包括A、B、C、D、E、F、G、DP
- 驱动方式：
 - (1) 静态驱动：每一个数码管引脚均有I/O独立控制，简单，亮度高
 - (2) 动态驱动：字段信号共用I/O，通过选通公共端轮流控制，节约资源，亮度低



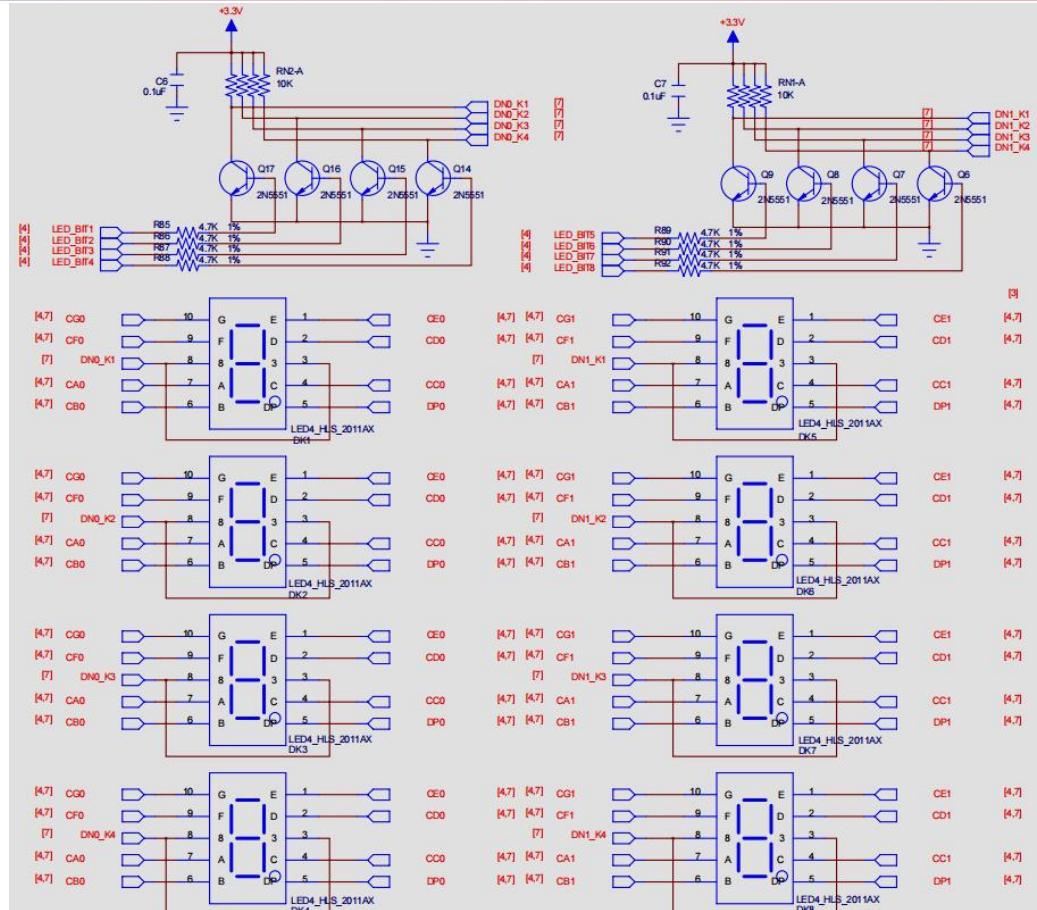
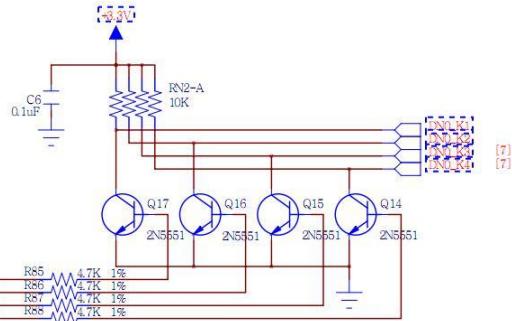
Lab4数码管实验1



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ EGo1数码管电路

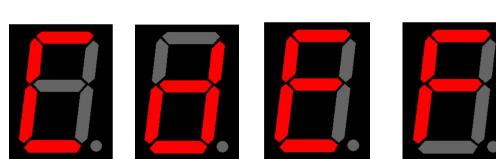
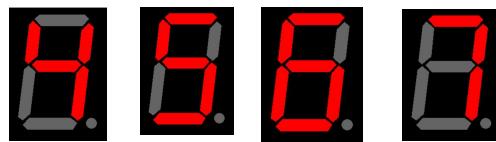
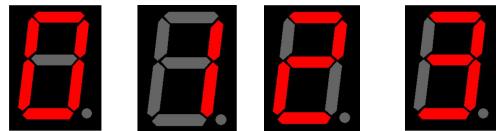
- 两组4位数码管，共8颗
- 共阴极设计 (LEDBITx为位选通信号，高电平有效)
- 第一组字段信号 (CA0/CB0/CC0/CD0/CE0/CF0/CG0/DP0)
- 第二组字段信号 (CA1/CB1/CC1/CD1/CE1/CF1/CG1/DP1)
- 字段信号高电平有效



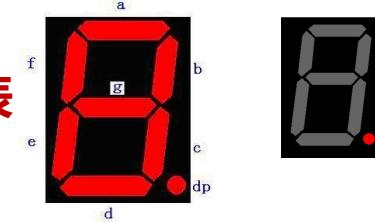
Lab4数码管实验1



國科大杭州高等研究院 Hangzhou Institute for Advanced Study, UCAS



字库表



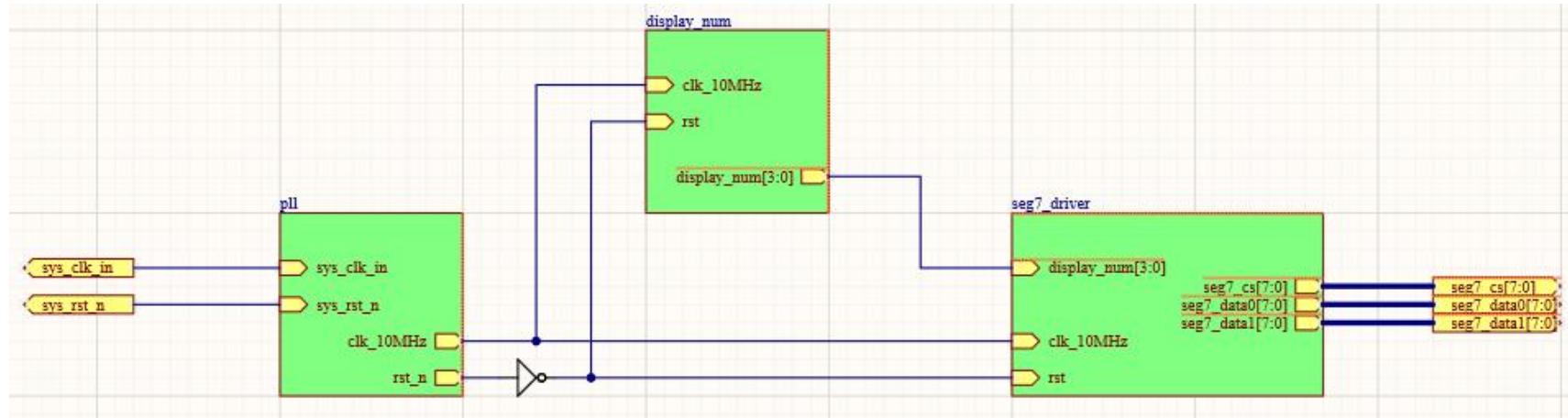


效果一 单颗数码管控制 实验步骤

Lab4数码管实验1



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



■ 系统框图

- 电路包括三个模块：pll模块实现时钟管理，display_num模块产生需要显示的数据，seg7_driver模块完成数码管驱动控制
- pll模块：通过IP核实现，100MHz时钟输入，全局复位引脚输入，生成10MHz信号，频率锁定信号用于后续模块复位
- display_num模块：10MHz时钟输入，频率锁定信号作为复位信号，输出需要显示的数据（每位数码管数据用4位表示）
- seg7_driver模块：10MHz时钟输入，频率锁定信号作为复位信号，通过seg7_cs,seg7_data控制数码管显示相应的数字



■ display_num模块

- 设计方案中计数的速度要可控制
- 注意计数频率，太快有可能导致无法看清

■ seg7_driver模块

- 定义数码管字库及位选控制信号
- 使用case语句对输入的display_num[3:0]信号进行查表，选择对应的字库输出
- 使用静态位选

Lab4数码管实验1



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

输入输出端口表

序号	信号名称	信号类型	输入/输出	信号描述	FPGA引脚	说明
1	sys_clk_in	1bit	输入	系统时钟输入	P17	100MHz输入
2	sys_RST_n	1bit	输入	系统复位输入	P15	低电平复位
3	seg7_cs[7:0]	8bit	输出	数码管位选控制信号输出	G6/E1/F1/G1 H1/C1/C2/G2	高电平选通
4	seg7_data0[7:0]	8bit	输出	数码管字段控制信号输出	D5/B2/B3/A1 B1/A3/A4/B4	高电平有效
5	seg7_data1[7:0]	8bit	输出	数码管字段控制信号输出	H2/D2/E2/F3 F4/D3/E3/D4	高电平有效

Lab4数码管实验1



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

模块输入输出端口表——锁相环模块 (pll)

序号	信号名称	信号类型	输入/输出	信号描述	说明
1	sys_clk_in	1bit	输入	系统时钟输入	100MHz输入
2	sys_RST_n	1bit	输入	系统复位输入	低电平复位
3	clk_10MHz	1bit	输出	系统主时钟	10MHz时钟输出
4	rst_n	1bit	输出	锁相环时钟锁定输出	高电平锁定

模块输入输出端口表——显示数据产生模块 (display_num)

序号	信号名称	信号类型	输入/输出	信号描述	说明
1	clk_10MHz	1bit	输入	系统主时钟	10MHz时钟输入
2	rst	1bit	输入	系统主复位	高电平复位
3	display_num	4bit	输出	显示数据输出	4bit, 0~F



模块输入输出端口表——数码管驱动模块 (seg7_driver)

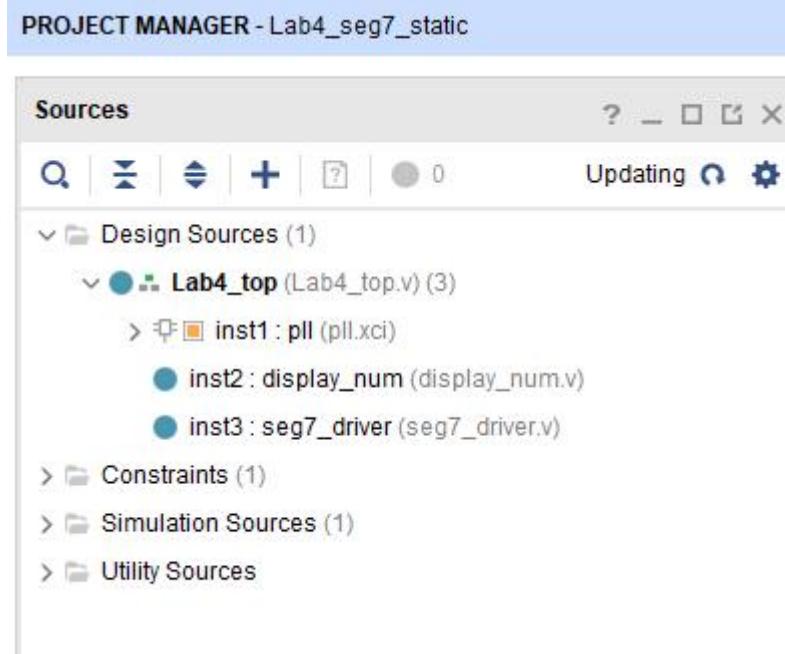
序号	信号名称	信号类型	输入/输出	信号描述	说明
1	clk_10MHz	1bit	输入	系统主时钟	10MHz时钟输入
2	rst	1bit	输入	系统主复位	高电平复位
3	display_num	4bit	输入	显示数据输入	4bit, 0~F
4	seg7_cs[7:0]	8bit	输出	数码管位选控制信号输出	高电平选通
5	seg7_data0[7:0]	8bit	输出	数码管字段控制信号输出	高电平有效
6	seg7_data1[7:0]	8bit	输出	数码管字段控制信号输出	高电平有效

Lab4数码管实验1



■ 实验步骤

- 建立工程
- 添加顶层设计文件, Lab4_top.v
- 通过IP catalog生成时钟模块pll.xci(inst1)
- 设计计数模块display_num.v(inst2)
- 设计数码管驱动模块seg7_driver.v(inst3)
- 在顶层文件完成模块例化及互联
- 添加引脚约束文件
- 设计仿真激励文件
- 完成功能仿真
- 综合
- 实现
- 下载调试



Lab4数码管实验1



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

```
module Lab4_top(
    input sys_clk_in,
    input sys_rst_n,
    output [7:0] seg7_cs,
    output [7:0] seg7_data0,
    output [7:0] seg7_data1
);

////////////////////////////////////////////////////////////////
wire clk_10MHz;
wire rst_n;
wire rst;
wire [3:0] display_num;

////////////////////////////////////////////////////////////////
assign rst = ~rst_n; // 将复位信号变成高电平有效

////////////////////////////////////////////////////////////////
pll inst1
(
    .clk_10MHz(clk_10MHz),          // output clk_10MHz
    .resetn(sys_rst_n),             // input resetn
    .rst_n(rst_n),                 // output rst_n
    .sys_clk_in(sys_clk_in)         // input sys_clk_in
);

////////////////////////////////////////////////////////////////
display_num inst2
(
    .clk_10MHz(clk_10MHz),          // input clk_10MHz
    .rst(rst),                     // input rst
    .display_num(display_num)       // output display_num
);

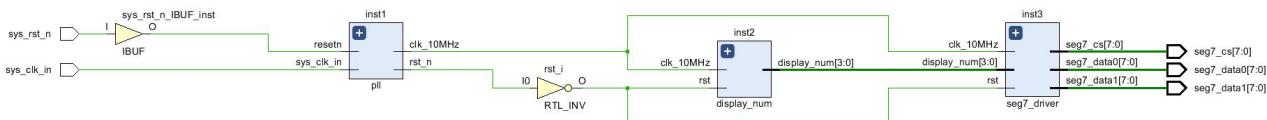
////////////////////////////////////////////////////////////////
seg7_driver inst3
(
    .clk_10MHz(clk_10MHz),          // input clk_10MHz
    .rst(rst),                     // input rst
    .display_num(display_num),      // input display_num
    .seg7_cs(seg7_cs),              // output seg7_cs
    .seg7_data0(seg7_data0),        // output seg7_data0
    .seg7_data1(seg7_data1)         // output seg7_data1
);

endmodule
```

顶层文件
做模块互联

关键代码

- 首先定义数码管字库及位选信号
- 根据输入数据从字库中进行索引输出
- 选择其中一颗数码管点亮
- 同一时刻所有数码管的字段信号均相同



Lab4数码管实验1



```
module display_num(
    input clk_10MHz,
    input rst,
    output reg [3:0] display_num
);

reg [31:0] counter;
parameter COUNTER_MAX = 10_000_000;

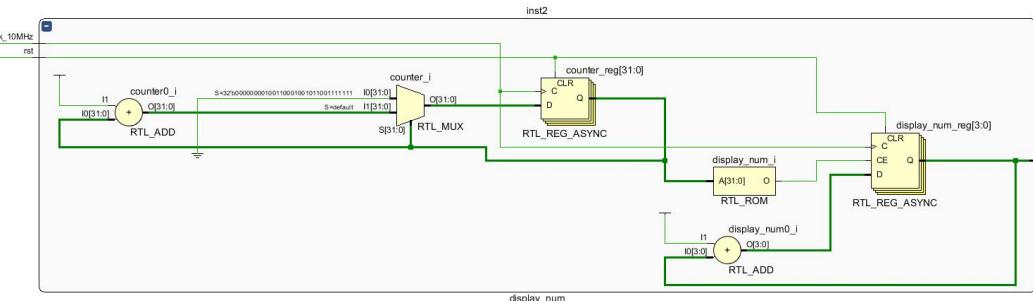
always @(posedge clk_10MHz or posedge rst) begin
    if(rst) begin
        counter <= 32'd0;
        display_num <= 4'd0;
    end
    else if(counter == (COUNTER_MAX-1)) begin
        counter <= 32'd0;
        display_num <= display_num + 1'b1;
    end
    else begin
        counter <= counter + 1'b1;
        display_num <= display_num;
    end
end

endmodule
```

显示数据产生模块

显示数据产生模块

1. 实现display_num从0~F的计数；
2. display_num的计数频率通过counter控制；
3. counter计数到COUNTER_MAX时display_num加1。



Lab4数码管实验1



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

```
module seg7_driver(
    input clk_10MHz,
    input rst,
    input [3:0] display_num,
    output reg [7:0] seg7_cs,
    output reg [7:0] seg7_data0,
    output reg [7:0] seg7_data1
);
```

字库

```
// 定义7段数码管字库
parameter NUM0 = 8'h3f,
        NUM1 = 8'h06,
        NUM2 = 8'h5b,
        NUM3 = 8'h4f,
        NUM4 = 8'h66,
        NUM5 = 8'h6d,
        NUM6 = 8'h7d,
        NUM7 = 8'h07,
        NUM8 = 8'h7f,
        NUM9 = 8'h6f,
        NUMA = 8'h77,
        NUMB = 8'h7c,
        NUMC = 8'h39,
        NUMD = 8'h5e,
        NUME = 8'h79,
        NUMF = 8'h71,
        NDOT = 8'h80;
```

位选

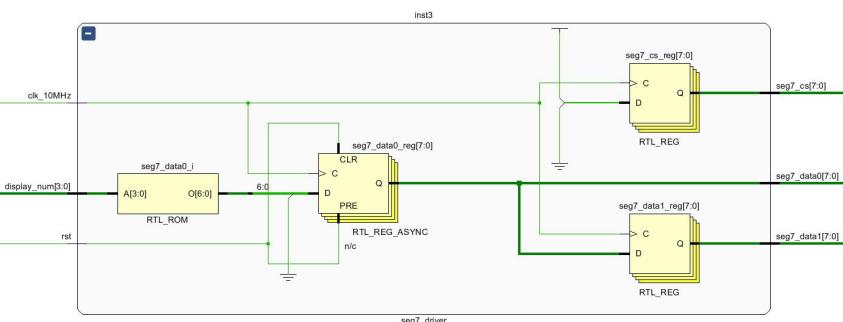
```
// 定义7段数码管选通信号
parameter CSN = 8'b0000_0000,
        CS0 = 8'b1000_0000,
        CS1 = 8'b0100_0000,
        CS2 = 8'b0010_0000,
        CS3 = 8'b0001_0000,
        CS4 = 8'b0000_1000,
        CS5 = 8'b0000_0100,
        CS6 = 8'b0000_0010,
        CS7 = 8'b0000_0001;
```

```
always @(posedge clk_10MHz or posedge rst) begin
    if(rst) begin
        seg7_data0 <= NUM0;
    end
    else begin
        case(display_num)
            4'h0: seg7_data0 <= NUM0;
            4'h1: seg7_data0 <= NUM1;
            4'h2: seg7_data0 <= NUM2;
            4'h3: seg7_data0 <= NUM3;
            4'h4: seg7_data0 <= NUM4;
            4'h5: seg7_data0 <= NUM5;
            4'h6: seg7_data0 <= NUM6;
            4'h7: seg7_data0 <= NUM7;
            4'h8: seg7_data0 <= NUM8;
            4'h9: seg7_data0 <= NUM9;
            4'ha: seg7_data0 <= NUMA;
            4'hb: seg7_data0 <= NUMB;
            4'hc: seg7_data0 <= NUMC;
            4'hd: seg7_data0 <= NUMD;
            4'he: seg7_data0 <= NUME;
            4'hf: seg7_data0 <= NUMF;
            default: seg7_data0 <= NUM0;
        endcase
    end
    seg7_cs <= CS0; //选择第0位点亮
    seg7_data1 <= seg7_data0;
endmodule
```

索引输出

数码管驱动模块

1. 定义字库及位选信号；
2. 根据输入的display_num数据，从字库索引输出对应的信号；
3. 选择一颗数码管进行点亮。

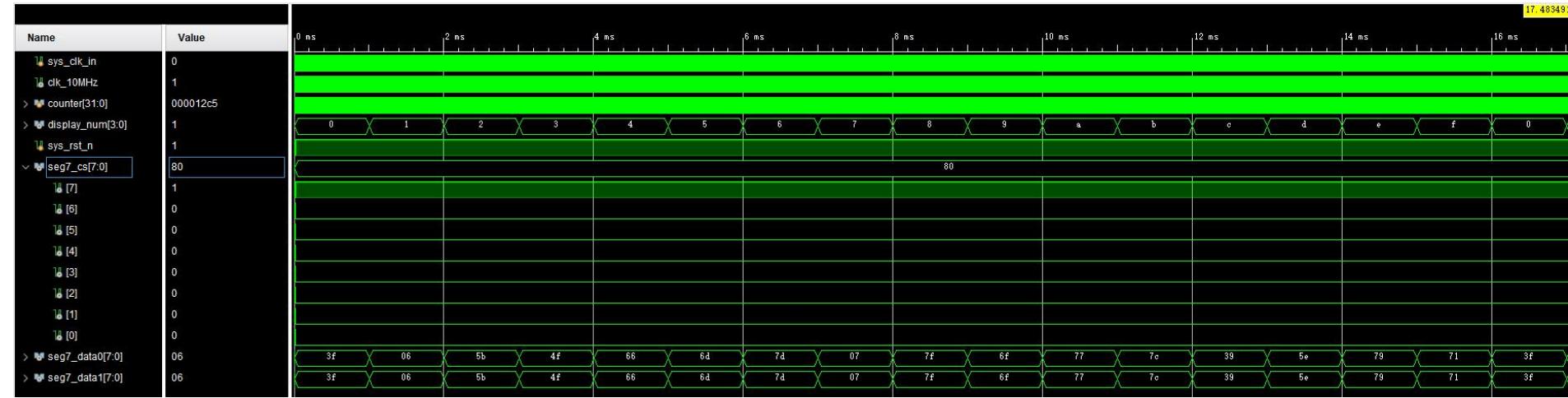


Lab4数码管实验1



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

功能仿真波形



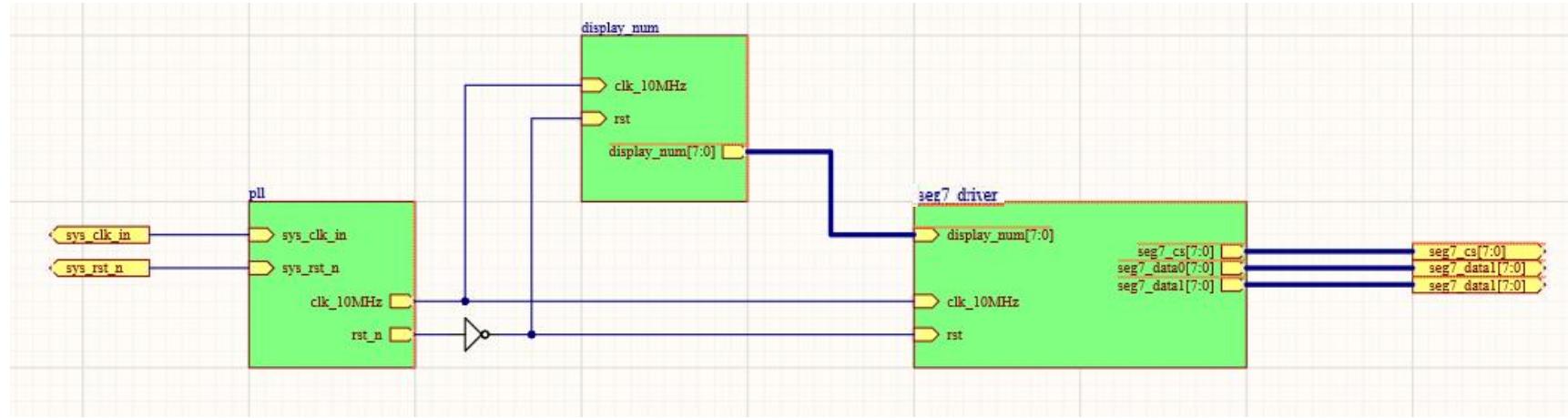


效果二
两颗数码管点亮
实现0~99计数

Lab4数码管实验1



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



■ 系统框图

- 电路包括三个模块：pll模块实现时钟管理，display_num模块产生需要显示的数据，seg7_driver模块完成数码管驱动控制
- pll模块：通过IP核实现，100MHz时钟输入，全局复位引脚输入，生成10MHz信号，频率锁定信号用于后续模块复位
- display_num模块：10MHz时钟输入，频率锁定信号作为复位信号，输出需要显示的数据（每位数码管数据用4位表示）
- seg7_driver模块：10MHz时钟输入，频率锁定信号作为复位信号，通过seg7_cs,seg7_data控制数码管显示相应的数字

Lab4数码管实验1



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

```
module display_num(
    input clk_10MHz,
    input rst,
    output reg [7:0] display_num
);

reg [31:0] counter;
parameter COUNTER_MAX = 10_000_000;

always @(posedge clk_10MHz or posedge rst) begin
    if(rst) begin
        counter <= 32'd0;
        display_num <= 4'd0;
    end
    else if(counter == (COUNTER_MAX-1)) begin
        counter <= 32'd0;
        if(display_num[3:0] == 4'd9) begin
            display_num[3:0] <= 4'd0;
            if(display_num[7:4] == 4'd9) begin
                display_num[7:4] <= 4'd0;
            end
            else begin
                display_num[7:4] <= display_num[7:4] + 1'b1;
            end
        end
        else begin
            display_num[3:0] <= display_num[3:0] + 1'b1;
        end
    end
    else begin
        counter <= counter + 1'b1;
        display_num <= display_num;
    end
end
endmodule
```

计数器模块(display_num.v)
产生显示所需的数据

按照1秒间隔进行计数

设计思路1：

注意两个切换节点：个位数为9；个位数与十位数同时为9。

设计流程：

1. 定义counter,对clk_10MHz时钟进行计数，产生秒计数标记；
2. 检测到秒标记 (counter == COUNTER_MAX-1) 后
如果低位计数值等于9，则：
 - (1) 将低位计数值清零；
 - (2) 如果高位计数值等于9，则：将高位计数值清零；
否则：高位计数值加1；
3. 没有到秒标记，则：counter加1

设计思路2：

1. 设计7bit计数器 (0x00~0x63)
2. 设计计数值拆分模块，拆分成两个4bit的数值

Lab4数码管实验1



```
module seg7_driver(
    input clk_10MHz,
    input rst,
    input [7:0] display_num,
    output reg [7:0] seg7_cs,
    output reg [7:0] seg7_data0,
    output reg [7:0] seg7_data1
);
```

// 定义7段数码管字库

```
parameter NUM0 = 8'h3f,
          NUM1 = 8'h06,
          NUM2 = 8'h5b,
          NUM3 = 8'h4f,
          NUM4 = 8'h66,
          NUM5 = 8'h6d,
          NUM6 = 8'h7d,
          NUM7 = 8'h07,
          NUM8 = 8'h7f,
          NUM9 = 8'h6f,
          NUMA = 8'h77,
          NUMB = 8'h7c,
          NUMC = 8'h39,
          NUMD = 8'h5e,
          NUME = 8'h79,
          NUMF = 8'h71,
          NDOT = 8'h80;
```

// 定义7段数码管选通信号

```
parameter CSN = 8'b0000_0000,
          CS0 = 8'b1000_0000,
          CS1 = 8'b0100_0000,
          CS2 = 8'b0010_0000,
          CS3 = 8'b0001_0000,
          CS4 = 8'b0000_1000,
          CS5 = 8'b0000_0100,
          CS6 = 8'b0000_0010,
          CS7 = 8'b0000_0001;
```

```
// 定义数码管刷新周期计数器
reg [31:0] cnt_refresh;
// 定义当前正在点亮的数码管编号，两颗数码管可以使用bit型进行切换
reg led_num;

// 定义数码管刷新间隔
parameter REFRESH_TIME = 100_000;

// 定义当前显示的数据
reg [7:0] current_display_num;

always @(posedge clk_10MHz or posedge rst) begin
    if(rst) begin
        cnt_refresh <= 32'd0;
        led_num <= 1'b0;
    end
    else if(cnt_refresh == (REFRESH_TIME -1)) begin
        cnt_refresh <= 32'd0;
        led_num <= ~led_num;
    end
    else begin
        cnt_refresh <= cnt_refresh + 1'b1;
    end
    current_display_num <= (led_num) ? display_num[7:4] : display_num[3:0];
    seg7_cs <= (led_num) ? CS1 : CS0;
end
```

数码管驱动模块：

设计思路：

- (1) 定义数码管刷新计数器，刷新周期10ms;
- (2) 定义数码管编号led_num;
- (3) 刷新周期达到后，翻转数码管编号，相当于数码管编号在0~1之间切换;
- (4)根据数码管编号输出字段及位段信号;
- (5)字库索引模块根据送入的当前显示数据进行索引输出

```
always @(posedge clk_10MHz or posedge rst) begin
    if(rst) begin
        seg7_data0 <= NUM0;
    end
    else begin
        case(current_display_num)
            4'h0: seg7_data0 <= NUM0;
            4'h1: seg7_data0 <= NUM1;
            4'h2: seg7_data0 <= NUM2;
            4'h3: seg7_data0 <= NUM3;
            4'h4: seg7_data0 <= NUM4;
            4'h5: seg7_data0 <= NUM5;
            4'h6: seg7_data0 <= NUM6;
            4'h7: seg7_data0 <= NUM7;
            4'h8: seg7_data0 <= NUM8;
            4'h9: seg7_data0 <= NUM9;
            4'ha: seg7_data0 <= NUMA;
            4'hb: seg7_data0 <= NUMB;
            4'hc: seg7_data0 <= NUMC;
            4'hd: seg7_data0 <= NUMD;
            4'he: seg7_data0 <= NUME;
            4'hf: seg7_data0 <= NUMF;
            default: seg7_data0 <= NUM0;
        endcase
    end
    seg7_data1 <= seg7_data0;
end

endmodule
```



附加实验
三颗数码管点亮
实现0~999计数

课后作业：HDLbits题目



- ▶ Getting Started
- ▶ Verilog Language
- ▼ Circuits
 - ▶ Combinational Logic
 - ▼ Sequential Logic
 - ▶ Latches and Flip-Flops
 - ▶ Counters
 - ▶ Shift Registers
 - ▶ More Circuits
 - ▼ Finite State Machines
 - Simple FSM 1
(asynchronous reset)
 - Simple FSM 1
(synchronous reset)
 - Simple FSM 2
(asynchronous reset)**
 - Simple FSM 2
(synchronous reset)
 - Simple state transitions 3
 - Simple one-hot state transitions 3



THANKS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第七讲 (7.1 状态机)

2025年4月15日



解决FPGA设计中使用状态机的核心问题：

为什么要用状态机？

什么是状态机？

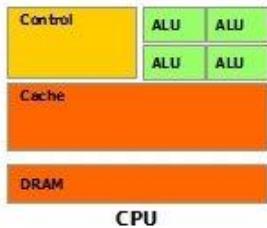
如何设计状态机？

如何使用状态机？

为什么要用状态机



X86架构计算数组相加



计算两个数组相加:

$\text{result}[i] = \text{array1}[i] + \text{array2}[i]$

for(int i = 0; i < 10; i++)
{
 result[i] = array1[i] + array2[i]; // 将两个数组对应位置的元素相加，并存储到 result 数组中
}

用c语言实现

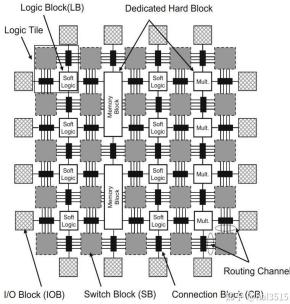
_start:
 mov ecx, 10 ; 设置循环计数器为 10 1个时钟周期
 mov esi, 0 ; 数组索引初始化为 0 1个时钟周期

add_loop:
 mov al, [array1 + esi] ; 从 array1 中取一个元素到 al 寄存器 1个时钟周期
 add al, [array2 + esi] ; 加上 array2 中对应位置的元素 2-3个时钟周期
 mov [result + esi], al ; 将结果存入 result 数组 1个时钟周期
 inc esi ; 数组索引加 1 1个时钟周期
 loop add_loop ; 循环，直到 ecx 为 0 3~5个时钟周期

用汇编语言实现

需要注意的是：上述时钟周期分析只是一个大致的估计，实际的指令周期数会因CPU型号、微架构、系统负载等因素而有所不同。
在现代CPU性能分析中，通常会使用专门的性能分析工具（如Intel VTune、Perf等）来准确测量指令的执行时间。

为什么要用状态机



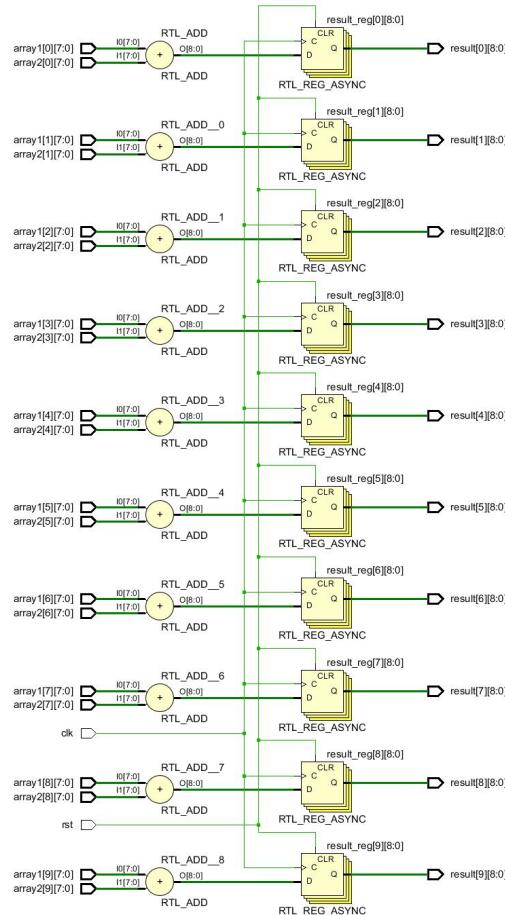
计算两个数组相加:

$\text{result}[i] = \text{array1}[i] + \text{array2}[i]$

FPGA的核心优势：硬件并行

```
// 使用generate块来展开循环
genvar i;
generate
    for (i = 0; i < 10; i = i + 1) begin : add_loop
        always @ (posedge clk or posedge rst) begin
            if (rst) begin
                // 复位时将结果数组对应元素清零
                result[i] <= 9'b0;
            end else begin
                // 正常时钟上升沿，执行数组对应元素相加操作
                result[i] <= array1[i] + array2[i];
            end
        end
    endgenerate
```

generate块用于在编译时展开循环
创建 10 个相同结构的 always 块
每个 always 块处理数组中的一个元素





为什么要用状态机

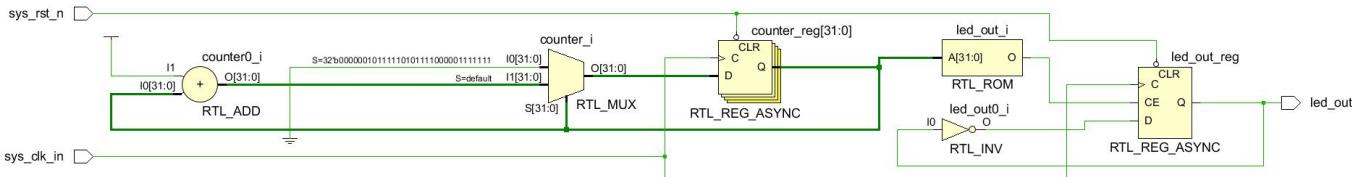
```
module lab2_counter(
    input sys_clk_in,
    input sys_rst_n,
    output reg led_out
);

// 定义计数器
reg [31:0] counter;

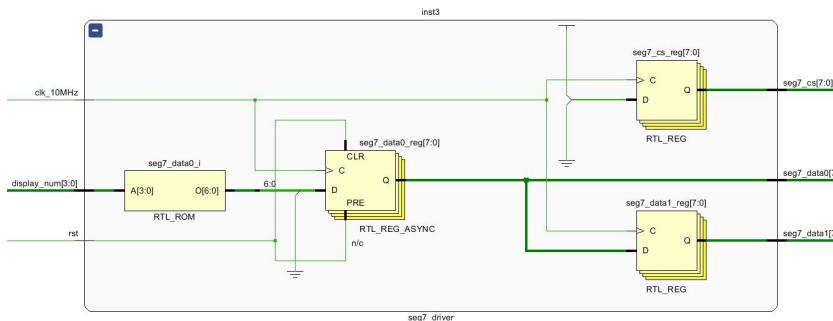
// 定义计数最大值，方便后续修改
parameter COUNTER_MAX = 32'd50_000_000;

// 设计一个时序电路--计数器
always @posedge sys_clk_in or negedge sys_rst_n) begin
    if(!sys_rst_n) begin
        counter <= 32'd0;
        led_out <= 1'b0;
    end
    // 计数到最大值时，将计数器清零，同时翻转led_out电平
    else if(counter == (COUNTER_MAX-1)) begin
        counter <= 32'd0;
        led_out <= ~led_out;
    end
    // 计数器在sys_clk_in上升沿时加1计数
    else
        counter <= counter + 1'b1;
end
endmodule

always @ (posedge clk_10MHz or posedge rst) begin
    if(rst) begin
        seg7_data0 <= NUM0;
    end
    else begin
        case(display_num)
            4'h0: seg7_data0 <= NUM0;
            4'h1: seg7_data0 <= NUM1;
            4'h2: seg7_data0 <= NUM2;
            4'h3: seg7_data0 <= NUM3;
            4'h4: seg7_data0 <= NUM4;
            4'h5: seg7_data0 <= NUM5;
            4'h6: seg7_data0 <= NUM6;
            4'h7: seg7_data0 <= NUM7;
            4'h8: seg7_data0 <= NUM8;
            4'h9: seg7_data0 <= NUM9;
            4'ha: seg7_data0 <= NUMA;
            4'hb: seg7_data0 <= NUMB;
            4'hc: seg7_data0 <= NUMC;
            4'hd: seg7_data0 <= NUMD;
            4'he: seg7_data0 <= NUME;
            4'hf: seg7_data0 <= NUMF;
            default: seg7_data0 <= NUM0;
        endcase
    end
    seg7_cs <= CS0;           // 选择第0位点亮
    seg7_data1 <= seg7_data0;
end
endmodule
```



FPGA时序电路的顺序控制：在时钟脉动下数据有序流动
痛点：复杂跳转控制逻辑不易实现



为什么要用状态机

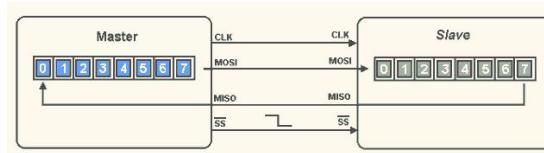
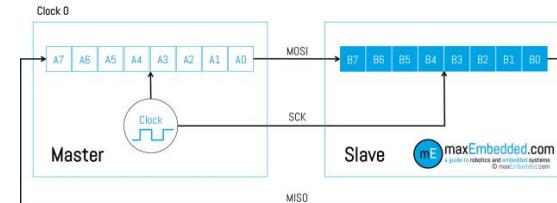


- 实际应用中，许多任务（如通信协议、数据流控制）需按特定时序或逻辑顺序执行
- CPU：容易实现逻辑顺序控制，但不容易保证时序精度
- FPGA：容易保证时序精度，但不容易实现逻辑顺序控制



FPGA引入状态机机制
加强逻辑顺序控制能力

SPI协议



Slave Select

Clock

MISO

MOSI

以SPI接收为例，设计4个状态：

- (1) IDLE: 等待传输开始，Slave Select高电平
- (2) START: 初始化传输，检测到Slave Select低电平
- (3) SHIFT: 在时钟控制下采集数据，MISO输出数据
- (4) STOP: 结束采集，数据校验，返回IDLE状态

为什么要用状态机



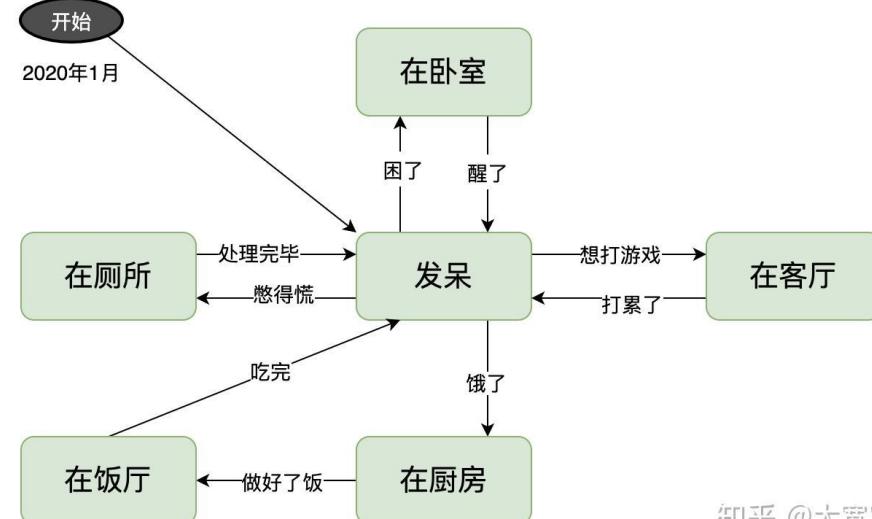
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

■为什么要用状态机？

- FPGA本身是硬件，具有天然的并行特性，既是优点也是缺点
- 应对控制系统的复杂逻辑性要求
- 状态机是高效的顺序控制模型
- 通过对状态的有序切换控制，实现复杂逻辑控制
- 使用状态机几乎可以实现一切控制电路
- 是FPGA的核心设计思想之一

■什么是状态机

- Finite State Machine(FSM)，是有限状态控制机的简称，是现实事物运行规律抽象而成的数学模型
- 状态机的本质是对具有逻辑顺序或时序规律事件的描述方法
- 有限状态机，被描述的事件的状态数量是有限的
- 状态机是各种程序设计中的通用思想



知乎 @大宽宽

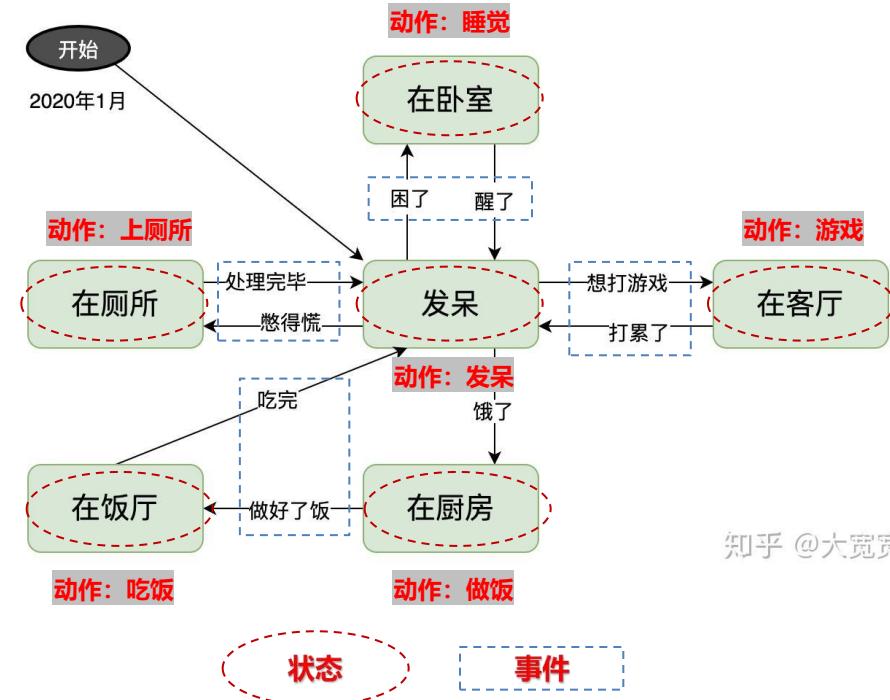
什么是状态机



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 状态机的组成

- **状态**: 状态机至少要包含两个状态
- **事件**: 触发条件, 能够触发状态机的状态发生改变
- **动作**: 在当前状态下要执行的事情
- **变换**: 从一个状态变换为另一个状态



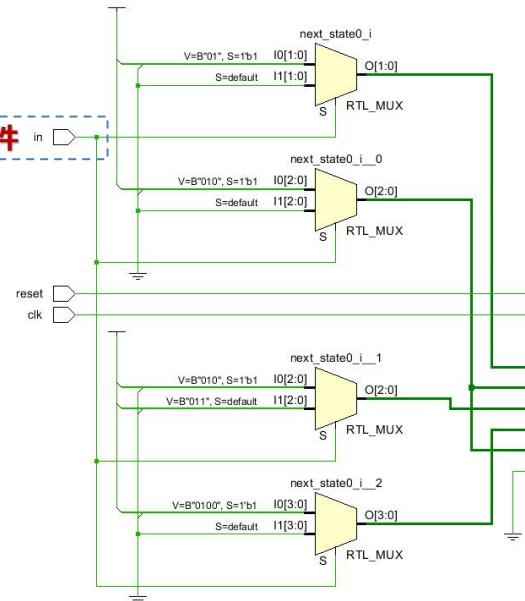
知乎 @大宽宽

什么是状态机

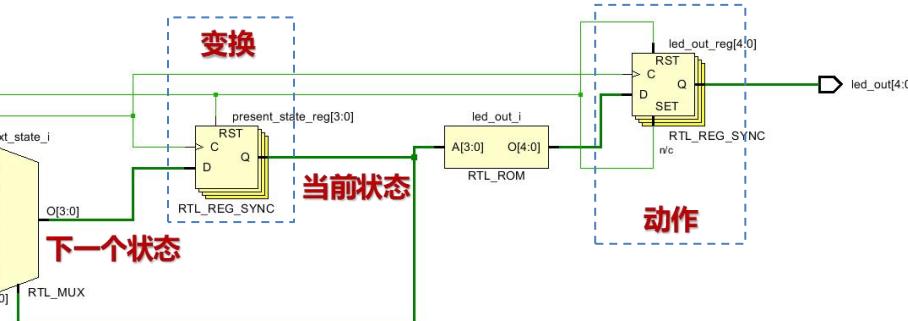


■用FPGA实现状态机的优点

- 性能稳定：状态机容易构成性能良好的同步时序逻辑电路
- 高速性能：在高速通信和高速控制方面，状态机具有巨大优势
- 高可靠性：状态机由纯硬件电路构成，运行不依赖软件指令的逐条执行，因此不存在CPU运行软件过程中许多固有缺陷（延迟、非实时、死机）



典型的FPGA状态机电路

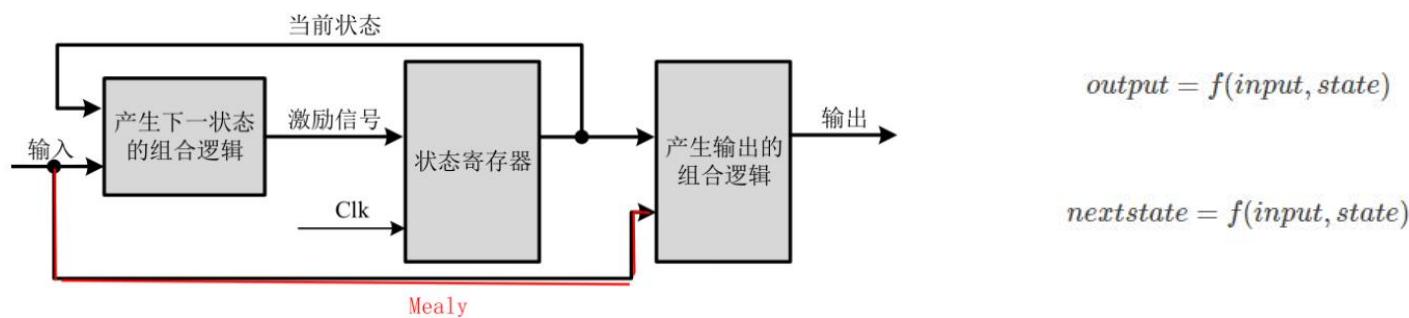
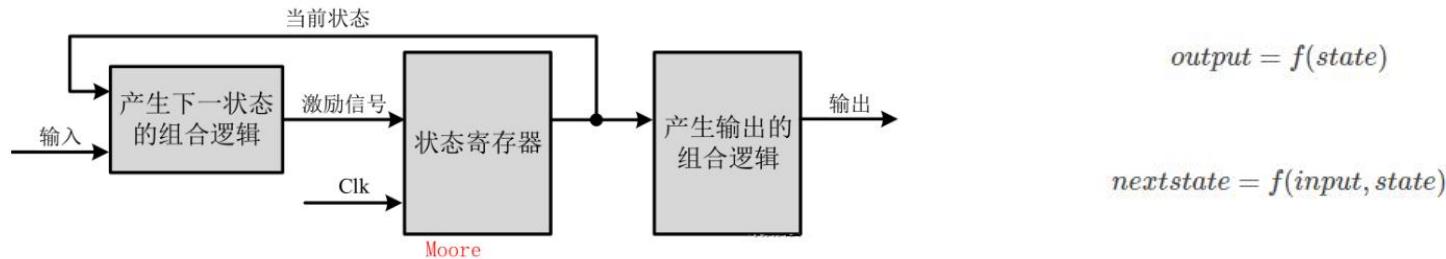




什么是状态机

■ 状态机的类型

- **Moore (摩尔) 型:** 输出只与当前状态有关, 与输入信号无关
- **Mealy (米勒) 型:** 输出与当前状态和输入均相关
- Moore型状态机对输入信号的响应有一个周期的延迟

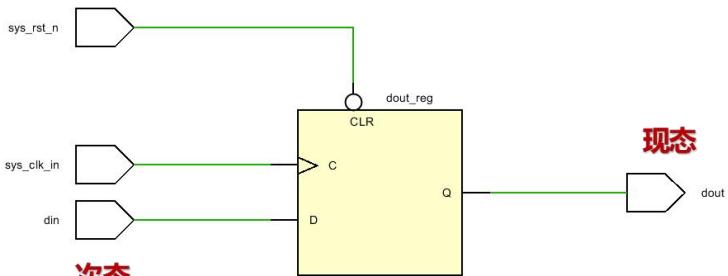


如何设计状态机

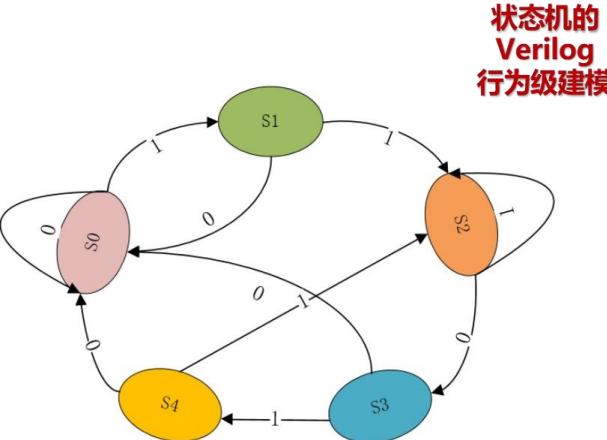


■ 状态图

- 描述状态机如何运转的图形表示
- 现态（状态）：状态机当前所处的状态
- 次态（状态）：状态机即将迁移到的状态
- 状态切换（变换）：一般由主时钟来控制切换，每次时钟边沿到达均会将次态变为现态
- 状态：S0、S1、S2、S3、S4
- 事件：现状态下进行判断，根据in信号的电平进行切换
- 动作：现状态下输出led_out信号



状态转换：时钟边沿到达后次态转换为现态



in信号为状态机状态转移控制信号；
led_out信号为状态机动作输出；

不同状态下信号输出示例：

S0: led_out <= 5' b00001
S1: led_out <= 5' b00011
S2: led_out <= 5' b00111
S3: led_out <= 5' b01111
S4: led_out <= 5' b11111

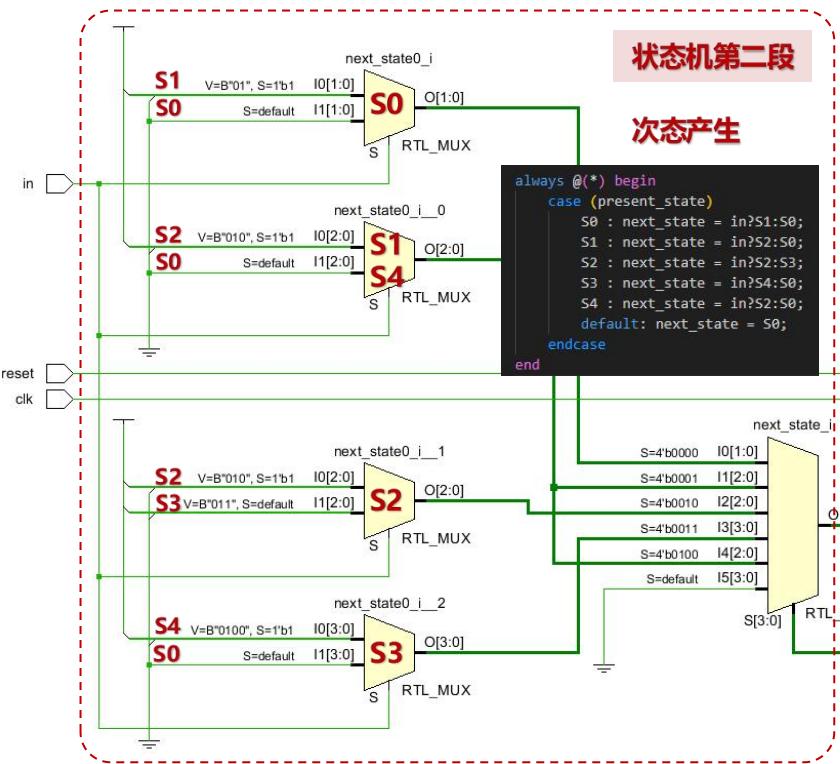
状态机的
Verilog
行为级建模

```
parameter S0 = 3'b000,  
        S1 = 3'b001,  
        S2 = 3'b010,  
        S3 = 3'b011,  
        S4 = 3'b100;  
  
reg [3:0] present_state, next_state;  
  
always @(posedge clk) begin  
    if(reset) begin  
        present_state <= S0; 复位  
    end  
    else begin  
        present_state <= next_state; 状态切换  
    end  
  
always @(*) begin  
    case (present_state)  
        S0 : next_state = in?S1:S0;  
        S1 : next_state = in?S2:S0;  
        S2 : next_state = in?S2:S3;  
        S3 : next_state = in?S4:S0;  
        S4 : next_state = in?S2:S0;  
        default: next_state = S0;  
    endcase  
end  
  
always @(posedge clk) begin  
    if(reset)begin  
        led_out <= 5'b00001;  
    end  
    else begin  
        case (present_state)  
            S0: led_out <= 5'b00001;  
            S1: led_out <= 5'b00011;  
            S2: led_out <= 5'b00111;  
            S3: led_out <= 5'b01111;  
            S4: led_out <= 5'b11111;  
            default: led_out <= 5'b00001;  
        endcase  
    end  
end
```

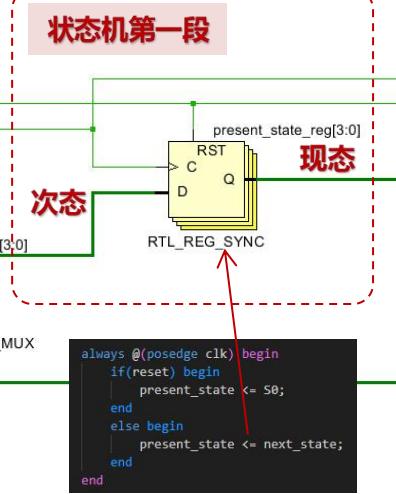
状态转移判断
次态产生
现状态下
动作输出



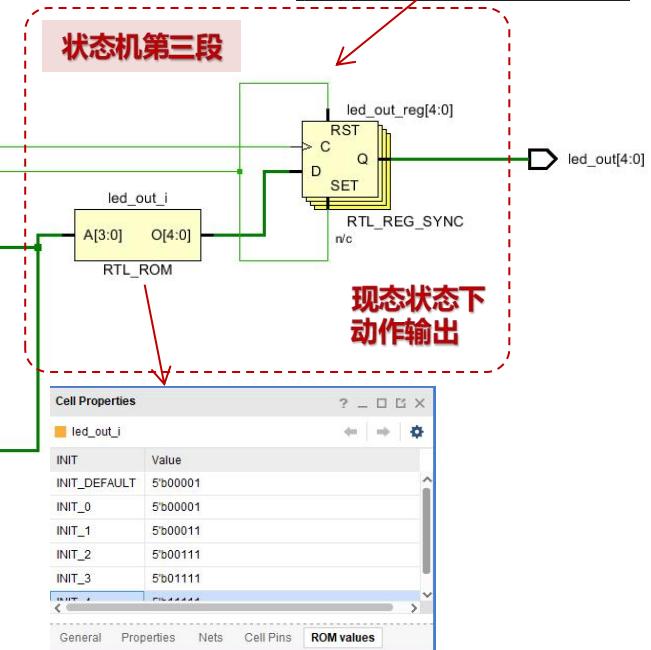
如何设计状态机



```
parameter S0 = 3'b000,
           S1 = 3'b001,
           S2 = 3'b010,
           S3 = 3'b011,
           S4 = 3'b100;
```



```
always @posedge clk begin
    if(reset) begin
        led_out <= 5'b00001;
    end
    else begin
        case (present_state)
            S0: led_out <= 5'b00001;
            S1: led_out <= 5'b00011;
            S2: led_out <= 5'b00111;
            S3: led_out <= 5'b01111;
            S4: led_out <= 5'b11111;
            default: led_out <= 5'b00001;
        endcase
    end
end
```

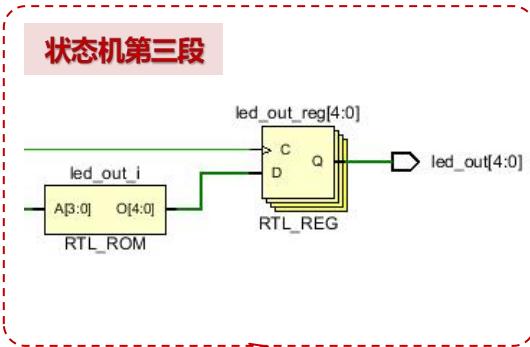


如何设计状态机

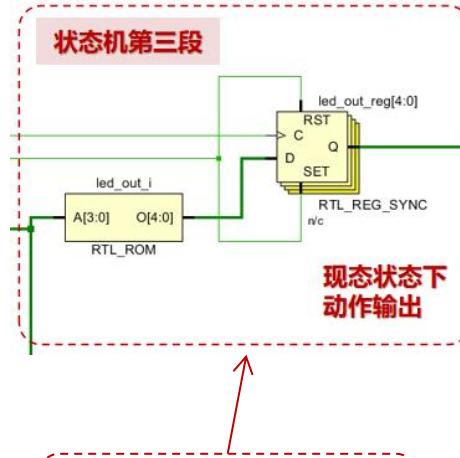


```
parameter S0 = 3'b000,  
        S1 = 3'b001,  
        S2 = 3'b010,  
        S3 = 3'b011,  
        S4 = 3'b100;  
  
reg [2:0] present_state, next_state;  
  
always @(posedge clk) begin  
    if(reset) begin  
        present_state <= S0;  
    end  
    else begin  
        present_state <= next_state;  
    end  
end  
  
always @(*) begin  
    case (present_state)  
        S0 : next_state = in?S1:S0;  
        S1 : next_state = in?S2:S0;  
        S2 : next_state = in?S2:S3;  
        S3 : next_state = in?S4:S0;  
        S4 : next_state = in?S2:S0;  
        default: next_state = S0;  
    endcase  
end  
  
always @(posedge clk) begin  
    case (present_state)  
        S0: led_out <= 5'b00001;  
        S1: led_out <= 5'b00011;  
        S2: led_out <= 5'b00111;  
        S3: led_out <= 5'b01111;  
        S4: led_out <= 5'b11111;  
        default: led_out <= 5'b00001;  
    endcase  
end
```

现态状态下动作输出
不加复位



现态状态下动作输出
加复位



```
always @(posedge clk) begin  
    if(reset)begin  
        led_out <= 5'b00001;  
    end  
    else begin  
        case (present_state)  
            S0: led_out <= 5'b00001;  
            S1: led_out <= 5'b00011;  
            S2: led_out <= 5'b00111;  
            S3: led_out <= 5'b01111;  
            S4: led_out <= 5'b11111;  
            default: led_out <= 5'b00001;  
        endcase  
    end
```

如何设计状态机



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 状态机写法

- 一段式：用一个always块来描述状态机，既描述状态转移，又描述状态的输入和输出
- 二段式：用两个always块来描述状态机，一个用同步时序模块描述状态转移，另一个用组合逻辑判断状态转移条件及输出
- 三段式：用三个always块来描述状态机，一个用同步时序模块描述状态转移，另一个用组合逻辑判断状态转移条件，再用一个同步时序模块描述状态输出

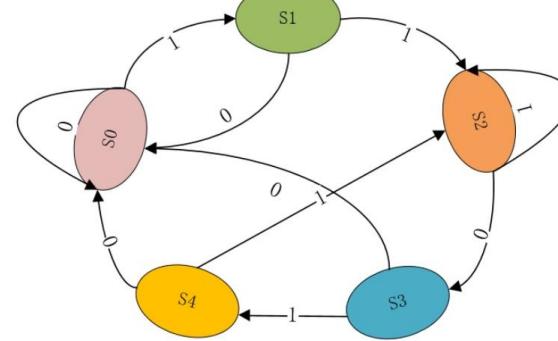
■ 三种状态机的特点

- 一段式代码不便于代码阅读、理解及维护
- 二段式使用组合逻辑实现输出，容易产生毛刺
- 推荐使用三段式状态机，三段式状态机采用时序逻辑进行输出

如何设计状态机



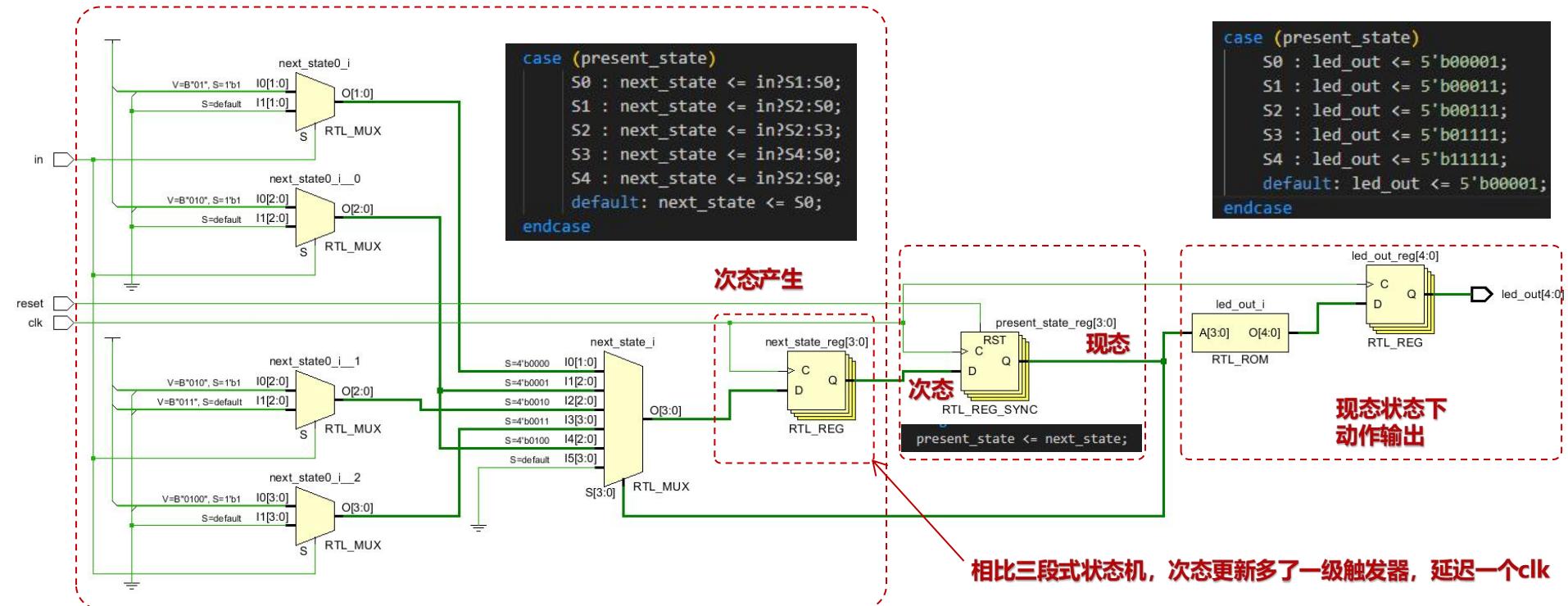
```
parameter S0 = 3'b000,  
S1 = 3'b001,  
S2 = 3'b010,  
S3 = 3'b011,  
S4 = 3'b100;  
  
reg [3:0] present_state, next_state;  
  
always @(posedge clk) begin  
    if(reset) begin  
        present_state <= S0; 复位  
    end  
    else begin  
        present_state <= next_state;  
    end  
    case (present_state) 状态切换  
        S0 : next_state <= in?S1:S0;  
        S1 : next_state <= in?S2:S0;  
        S2 : next_state <= in?S3:S0; 状态转移判断  
        S3 : next_state <= in?S4:S0; 次态产生  
        S4 : next_state <= in?S2:S0;  
        default: next_state <= S0;  
    endcase  
    case (present_state)  
        S0 : led_out <= 5'b00001;  
        S1 : led_out <= 5'b00011;  
        S2 : led_out <= 5'b00111;  
        S3 : led_out <= 5'b01111;  
        S4 : led_out <= 5'b11111;  
        default: led_out <= 5'b00001;  
    endcase  
end
```



■一段式状态机

- 状态: S0,S1,S2,S3,S4, 同步复位时状态为S0
- 状态切换、状态转移判断、现态下状态输出全部写为一段

如何设计状态机



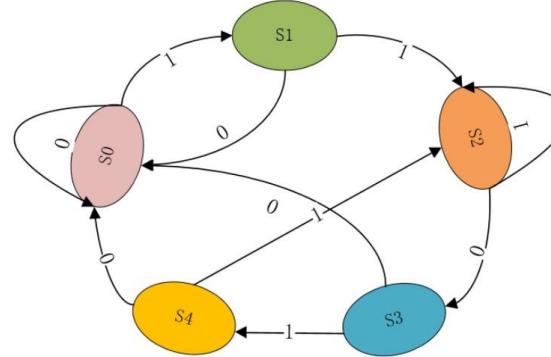
如何设计状态机



```
parameter S0 = 3'b000,  
S1 = 3'b001,  
S2 = 3'b010,  
S3 = 3'b011,  
S4 = 3'b100;  
  
reg [3:0] present_state, next_state;  
  
always @ (posedge clk) begin  
    if(reset) begin  
        present_state <= S0; 复位  
    end  
    else begin  
        present_state <= next_state;  
    end  
end  
  
always @(*) begin  
    case (present_state)  
        S0 : begin  
            led_out = 5'b00001;  
            next_state = in?S1:S0;  
        end  
        S1 : begin  
            led_out = 5'b00011;  
            next_state = in?S2:S0;  
        end  
        S2 : begin  
            led_out = 5'b00111;  
            next_state = in?S3:S3;  
        end  
        S3 : begin  
            led_out = 5'b01111;  
            next_state = in?S4:S0;  
        end  
        S4 : begin  
            led_out = 5'b11111;  
            next_state = in?S2:S0;  
        end  
        default: begin  
            led_out = 5'b00001;  
            next_state = S0;  
        end  
    endcase  
end
```

状态切换

状态转移判断（次态产生）
现态状态下动作输出



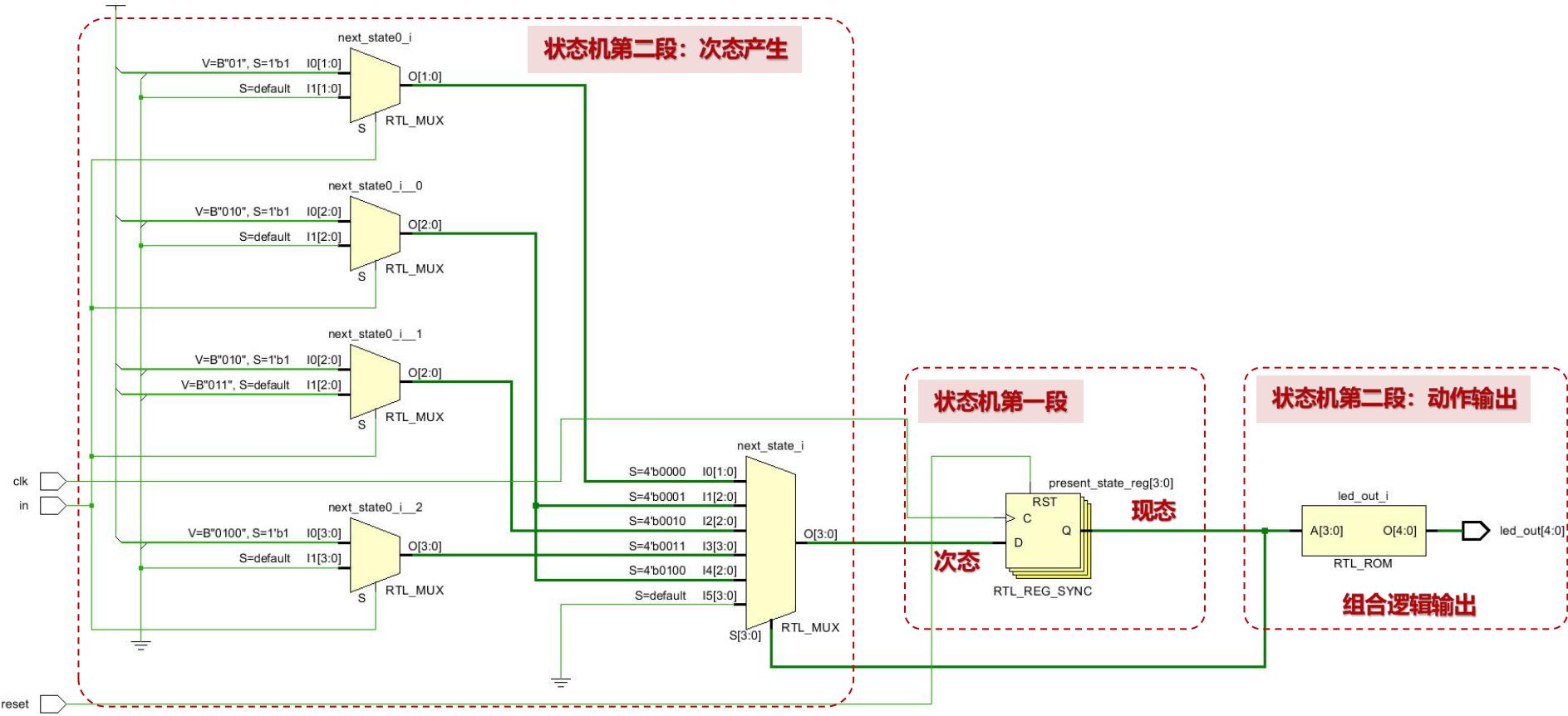
■二段式状态机

- 状态: S0,S1,S2,S3,S4, 同步复位时状态为S0
- 第一段: 时序逻辑, 实现复位及状态转换
- 第二段: 组合逻辑, 实现状态转移判断及现态动作输出

如何设计状态机



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS



如何设计状态机

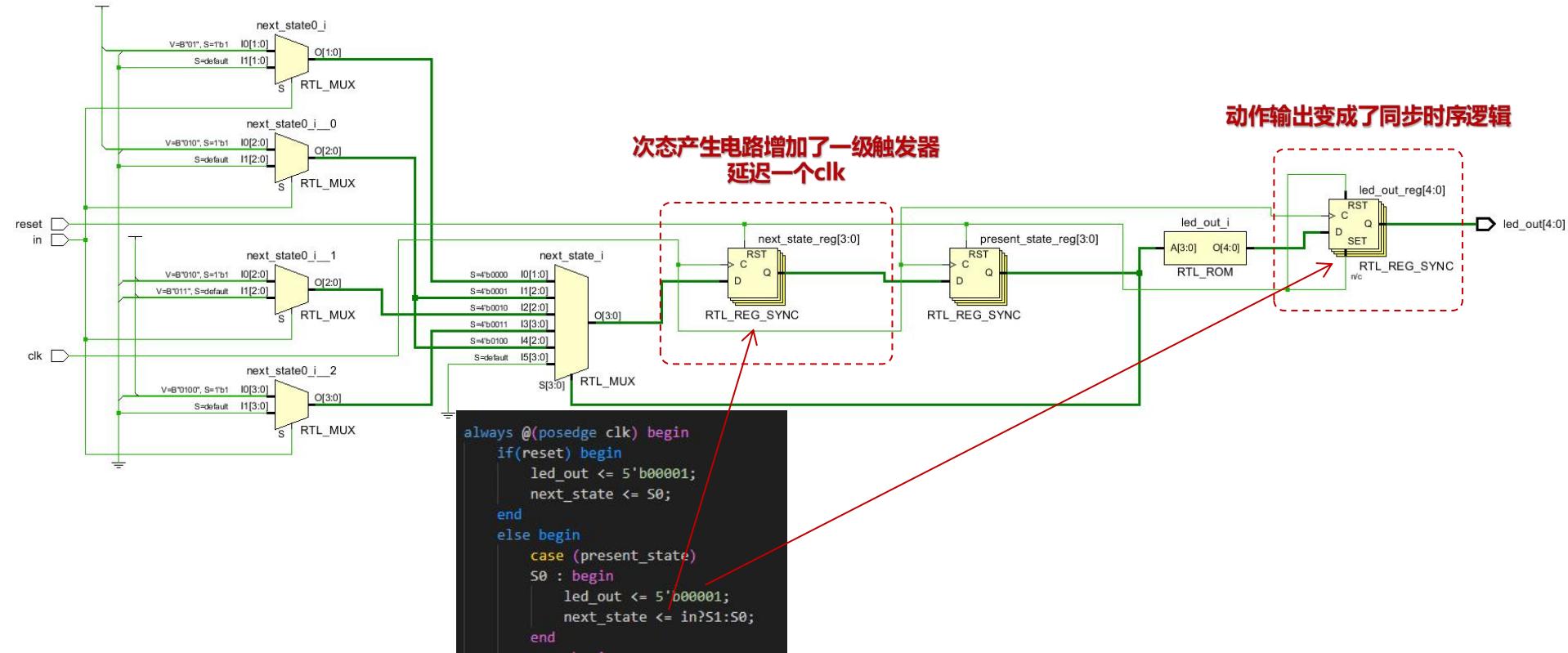


```
always @(posedge clk) begin
    if(reset) begin
        present_state <= S0;
    end
    else begin
        present_state <= next_state;
    end
end

always @(posedge clk) begin
    if(reset) begin
        led_out <= 5'b00001;
        next_state <= S0;
    end
    else begin
        case (present_state)
            S0 : begin
                led_out <= 5'b00001;
                next_state <= in?S1:S0;
            end
            S1 : begin
                led_out <= 5'b00011;
                next_state <= in?S2:S0;
            end
            S2 : begin
                led_out <= 5'b00111;
                next_state <= in?S3:S0;
            end
            S3 : begin
                led_out <= 5'b01111;
                next_state <= in?S4:S0;
            end
            S4 : begin
                led_out <= 5'b11111;
                next_state <= in?S2:S0;
            end
            default: begin
                led_out <= 5'b00001;
                next_state <= S0;
            end
        endcase
    end
end
```

两段式状态机的第二段改为时序逻辑?

如何设计状态机





推荐使用三段式状态机

逻辑清晰、维护简便

稳定性高、实时性好

如何设计状态机



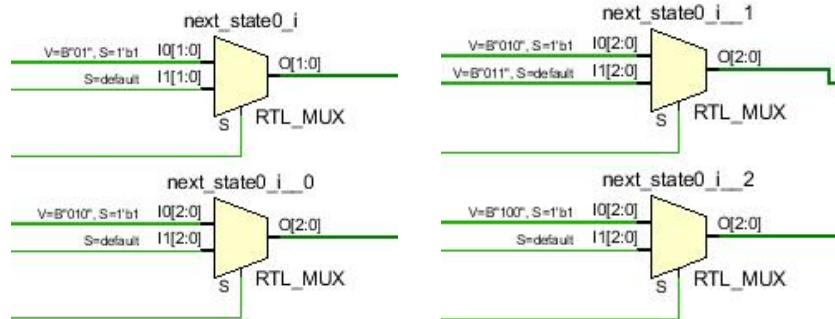
■ 状态编码

- 二进制编码**: 减少编码时触发器的数量，编码方式较为容易理解，使用较多组合逻辑，状态多时容易产生竞争与冒险
- 独热码**: 每一个状态均使用一个寄存器，在于状态比较时仅仅需要比较一位，节省组合逻辑资源，位宽多需要占用更多寄存器资源
- 格雷码**: 相邻位只跳动一位，是二进制编码与独热码的折衷
- 电路中具体的编码格式可通过**编译器**进行设置

状态	二进制	独热码	格雷码
A	3'b000	8'b0000_0000	4'b0000
B	3'b001	8'b0000_0010	4'b0001
C	3'b010	8'b0000_0100	4'b0011
D	3'b011	8'b0000_1000	4'b0010
E	3'b100	8'b0001_0000	4'b0110
F	3'b101	8'b0010_0000	4'b0111
G	3'b110	8'b0100_0000	4'b0101
H	3'b111	8'b1000_0000	4'b0100

Options

-gated_clock_conversion	off
-bufg	12
-fanout_limit	10,000
-directive	Default
-retiming	
-fsm_extraction	auto
-keep_equivalent_registers	off
-resource_sharing	one_hot
-control_set_opt_threshold	sequential
-no_lc	johson
-no_srlextract	gray



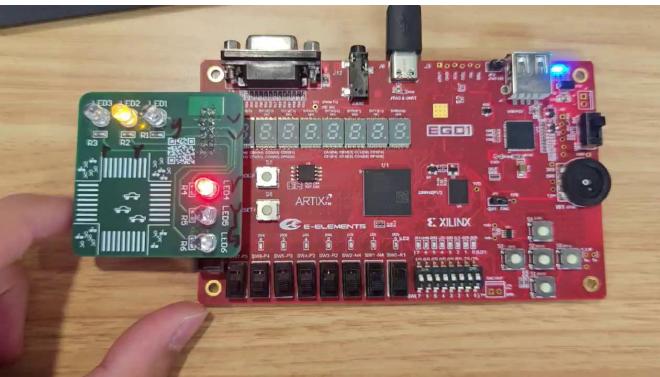
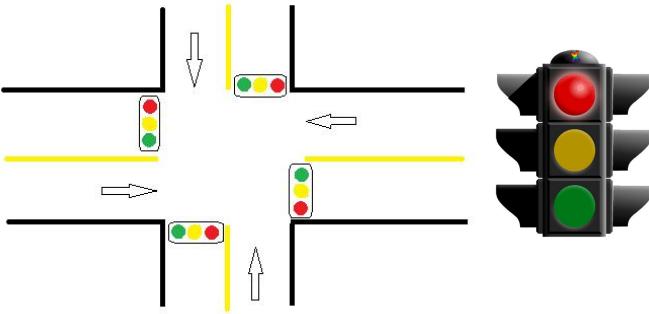
如何使用状态机-Lab5:交通灯



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 实验内容

- 控制两路红绿灯按照交通规则点亮
- 效果1：简易红绿灯
- [状态S0-复位]：复位状态下，横向及纵向所有灯全亮；
- [状态S1-复位释放后切换]：复位释放后，所有灯闪烁一次（1s）后进入状态S2；
- [状态S2-横向通行]：横向绿灯亮起（3s），纵向保持红灯亮起；
- [状态S3-横向切换纵向]：横向黄灯亮起（保持3秒），纵向保持红灯亮起；
- [状态S4-纵向通行]：纵向绿灯亮起（3秒），横向同时红灯亮起；
- [状态S5-纵向切换横向]：纵向黄灯亮起（保持3秒），横向保持红灯亮起；
- 返回状态2，保持状态2~状态5间循环，复位键按下后回到状态S0；



如何使用状态机-Lab5:交通灯

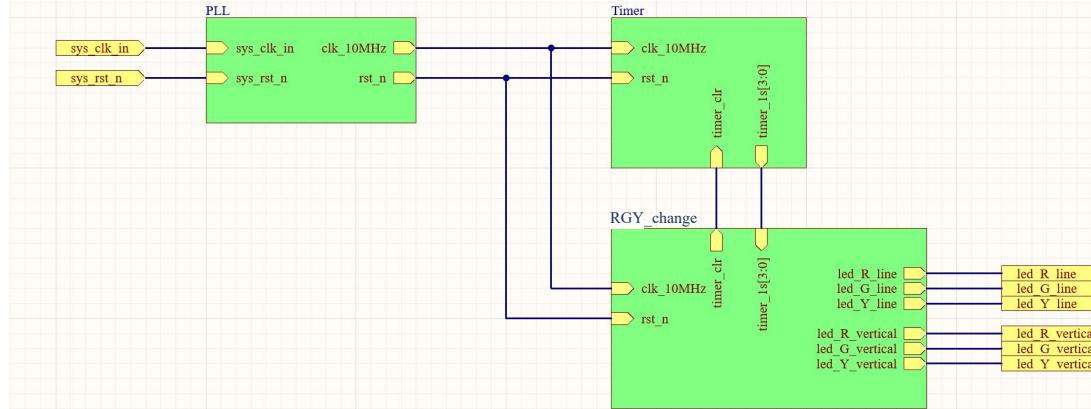


■ 效果2：修改红绿灯配时，增加优先通行功能

- [状态S0-复位]：复位状态下，横向及纵向所有灯全亮；
- [状态S1-复位释放后切换]：复位释放后，横向红灯闪烁3次 (1Hz) ->横向黄灯亮起 (保持3秒)，纵向保持红灯亮起；
- [状态S2-横向通行]：横向绿灯亮起 (15秒)，纵向保持红灯亮起；
 检测到横向优先通行，跳转到状态S6；检测到纵向优先通行，跳转到状态S7；
- [状态S3-横向切换纵向]：横向绿灯闪烁3秒 (1Hz) ->横向黄灯亮起 (保持3秒)，纵向保持红灯亮起；
- [状态S4-纵向通行]：纵向绿灯亮起 (15秒)，横向同时红灯亮起；
 检测到纵向优先通行，跳转到状态S7；检测到横向优先通行，跳转到状态S6；
- [状态S5-纵向切换横向]：纵向绿灯闪烁3秒 (1Hz) ->纵向黄灯亮起 (保持3秒)，横向保持红灯亮起；
- 返回状态S2，保持状态S2~状态S5间循环，复位键按下后回到状态S0；
- [状态S6-横向优先]：横向绿灯常亮，纵向红灯常亮；
 检测到退出横向优先通行，跳转到状态S3，否则保持状态S6；
- [状态S7-纵向优先]：横向红灯常亮，纵向绿灯常亮；
 检测到退出纵向优先通行，跳转到状态S5，否则保持状态S7；



如何使用状态机-Lab5:交通灯



■ 系统框图

- 电路包括三个模块：pll模块实现时钟管理，Timer模块实现带有清零功能的秒计数器，RGY_change模块进行红绿灯转换
- pll模块：通过IP核实现，100MHz时钟输入，全局复位引脚输入，生成10MHz信号，频率锁定信号用于后续模块复位
- Timer模块：一个带有清零功能的秒计数器，每次状态切换都会重新开始计时
- RGY_change模块：10MHz时钟输入，频率锁定信号作为复位信号，输出不同状态下的红绿灯的状态

如何使用状态机-Lab5:交通灯



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

输入输出端口表

序号	信号名称	信号类型	输入/输出	信号描述	FPGA引脚	说明
1	sys_clk_in	1bit	输入	系统时钟输入	P17	100MHz输入
2	sys_RST_n	1bit	输入	系统复位输入	P15	低电平复位
3	led_R_line	1bit	输出	横向红灯	H17	高电平有效
4	led_Y_line	1bit	输出	横向黄灯	K13	高电平有效
5	led_G_line	1bit	输出	横向绿灯	E17	高电平有效
6	led_R_vertical	1bit	输出	纵向红灯	D17	高电平有效
7	led_Y_vertical	1bit	输出	纵向黄灯	J13	高电平有效
8	led_G_vertical	1bit	输出	纵向绿灯	G17	高电平有效

如何使用状态机-Lab5:交通灯



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

输入输出端口表——锁相环模块 (pll)

序号	信号名称	信号类型	输入/输出	信号描述	说明
1	sys_clk_in	1bit	输入	系统时钟输入	100MHz输入
2	sys_RST_n	1bit	输入	系统复位输入	低电平复位
3	clk_10MHz	1bit	输出	10MHz时钟输出	10MHz输出
4	rst_n	1bit	输出	频率锁定信号	低电平复位

输入输出端口表——计数器模块 (Timer)

序号	信号名称	信号类型	输入/输出	信号描述	说明
1	clk_10MHz	1bit	输入	10MHz时钟输入	10MHz输入
2	rst_n	1bit	输入	系统复位输入	低电平复位
3	timer_clr	1bit	输入	计数清零信号	高电平有效
4	timer_1s	8bit	输出	1秒信号	输出8位信号

如何使用状态机-Lab5:交通灯



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

输入输出端口表——红绿灯状态转换模块 (RGY_change)

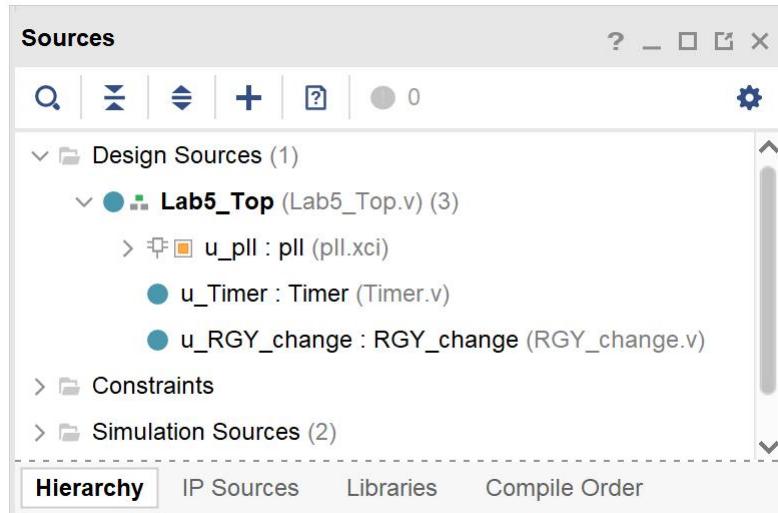
序号	信号名称	信号类型	输入/输出	信号描述	说明
1	clk_10MHz	1bit	输入	时钟信号	10MHz时钟
2	rst_n	1bit	输入	复位信号	低电平复位
3	timer_1s	8bit	输入	1秒信号	输出8位信号
4	timer_clr	1bit	输出	计数清零信号	高电平有效
5	led_R_line	1bit	输出	横向红灯	高电平有效
6	led_Y_line	1bit	输出	横向黄灯	高电平有效
7	led_G_line	1bit	输出	横向绿灯	高电平有效
8	led_R_vertical	1bit	输出	纵向红灯	高电平有效
9	led_Y_vertical	1bit	输出	纵向黄灯	高电平有效
10	led_R_vertical	1bit	输出	纵向红灯	高电平有效

如何使用状态机-Lab5:交通灯



■ 实验步骤

- 建立工程
- 添加顶层设计文件, lab5_Top.v
- 通过IP catalog生成时钟模块pll.xci(u_pll)
- 设计计数器模块Timer (u_Timer)
- 设计红绿灯状态转换模块
RGY_change(u_RGY_change)
- 在顶层文件完成模块例化及互联
- 添加引脚约束文件
- 设计仿真激励文件
- 完成功能仿真
- 综合
- 实现
- 下载调试



如何使用状态机-Lab5:交通灯



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

```
module Lab5_Top()
    input sys_clk_in,
    input sys_rst_n,
    output led_R_line,
    output led_G_line,
    output led_Y_line,
    output led_R_vertical,
    output led_G_vertical,
    output led_Y_vertical
);

////////////////////////////////////////////////////////////////// 例化pll时钟模块 //////////////////////////////////////////////////////////////////

wire rst_n;
wire clk_10MHz;
wire timer_clr;
wire [7:0] timer_ls;

////////////////////////////////////////////////////////////////// 例化pll时钟模块 //////////////////////////////////////////////////////////////////

pll u_pll (
    .clk_10MHz (clk_10MHz), // output clk_10MHz
    .resetn (sys_rst_n), // input resetn
    .locked (rst_n), // output locked
    .sys_clk_in(sys_clk_in)
);

////////////////////////////////////////////////////////////////// 例化计数器模块 //////////////////////////////////////////////////////////////////

Timer u_Timer(
    .clk_10MHz (clk_10MHz),
    .rst_n (rst_n),
    .timer_clr (timer_clr),
    .timer_ls (timer_ls)
);

////////////////////////////////////////////////////////////////// 例化红绿灯状态转换模块 //////////////////////////////////////////////////////////////////

RGY_change u_RGY_change (
    .clk_10MHz (clk_10MHz),
    .rst_n (rst_n),
    .timer_clr (timer_clr),
    .timer_ls (timer_ls),
    .led_R_line (led_R_line),
    .led_G_line (led_G_line),
    .led_Y_line (led_Y_line),
    .led_R_vertical(led_R_vertical),
    .led_G_vertical(led_G_vertical),
    .led_Y_vertical(led_Y_vertical)
);
```

顶层文件
做模块互联

关键代码

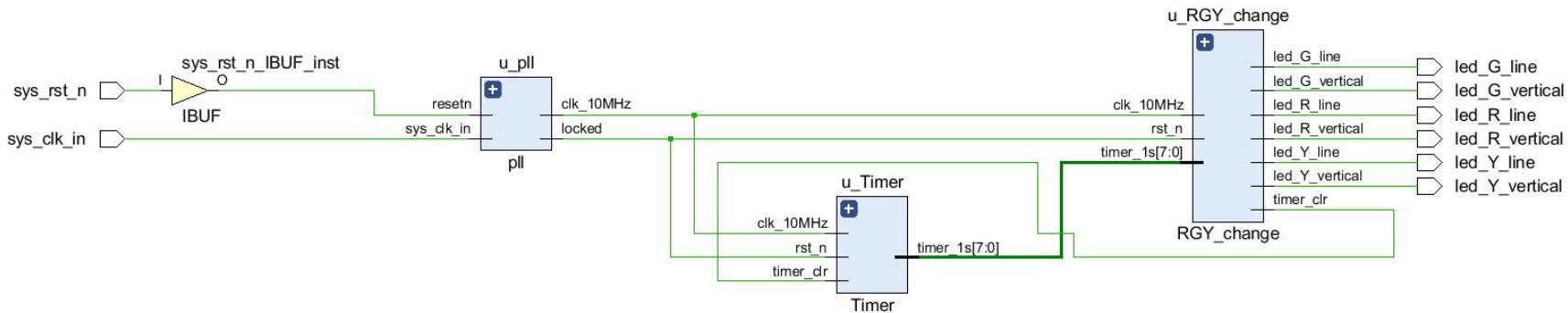
1. 例化pll时钟， Timer计数器模块以及红绿灯转换模块RGY_change，在顶层完成模块互联；
2. Timer是一个带有清零功能的秒计数器，每次状态切换都会重新开始计时；
3. 使用三段式状态机编写红绿灯状态转换模块；

如何使用状态机-Lab5:交通灯



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

顶层文件 (Lab5_TOP) 完成模块互联



如何使用状态机-Lab5:交通灯



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

```
module Timer(
    input clk_10MHz ,
    input rst_n ,
    input timer_clr ,
    output reg [3:0] timer_1s
)
parameter counter_MAX=100 ;
reg [31:0] counter ;
//////////////////计时器/////////////////
always@(posedge clk_10MHz or negedge rst_n)begin
    if(!rst_n)begin
        counter <= 32'b0 ;
        timer_1s <= 3'b0 ;
    end
    //状态切换时计数器清零
    else if (timer_clr) begin
        counter <= 32'b0 ;
        timer_1s <= 3'b0 ;
    end
    //每1S输出一次timer_1s信号
    else if (counter == counter_MAX-1) begin
        counter <= 32'b0 ;
        timer_1s <= timer_1s + 1'b1 ;
    end
    //计数器循环计数
    else begin
        counter<=counter+1'b1 ;
        timer_1s<= timer_1s ;
    end
end
endmodule
```

计数器模块Timer (u_Timer)

Timer是一个带有清零功能的秒计数器，每次状态切换都会重新开始计时

如何使用状态机-Lab5:交通灯



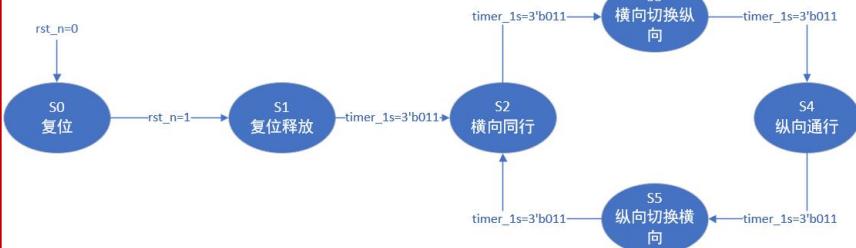
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

红绿灯状态转换模块

此模块分为6个状态，状态转换图如右图所示

- [状态S0-复位]: 复位状态下，横向及纵向所有灯全亮；
- [状态S1-复位释放后切换]: 复位释放后，所有灯闪烁一次 (1s) 后进入状态S2；
- [状态S2-横向通行]: 横向绿灯亮起 (3s)，纵向保持红灯亮起；
- [状态S3-横向切换纵向]: 横向黄灯亮起 (保持3秒)，纵向保持红灯亮起；
- [状态S4-纵向通行]: 纵向绿灯亮起 (3秒)，横向同时红灯亮起；
- [状态S5-纵向切换横向]: 纵向黄灯亮起 (保持3秒)，横向保持红灯亮起；
- 返回状态2，保持状态2~状态5间循环，复位键按下后回到状态S0；

```
s0 = 3'b000, //复位
s1 = 3'b001, //复位释放后切换
s2 = 3'b010, //横向通行
s3 = 3'b011, //横向切换纵向
s4 = 3'b100, //纵向通行
s5 = 3'b101; //纵向切换横向
```



状态切换条件:
S0: rst_n信号
其余状态按照顺序切换，切换条件是timer_1s的计数值

如何使用状态机-Lab5:交通灯



红绿灯状态转换模块

```
module RGY_change(
    input clk_10MHz,
    input rst_n,
    input [3:0] timer_1s,
    output timer_clr,
    output reg led_R_line,
    output reg led_G_line,
    output reg led_Y_line,
    output reg led_R_vertical,
    output reg led_G_vertical,
    output reg led_Y_vertical
);
    reg [2:0] persent_state, next_state;
    parameter
        s0 = 3'b000, //复位
        s1 = 3'b001, //复位释放后切换
        s2 = 3'b010, //横向通行
        s3 = 3'b011, //横向切换纵向
        s4 = 3'b100, //纵向通行
        s5 = 3'b101; //纵向切换横向
    //计数器清零信号//////状态转换时清零//////
    assign timer_clr = persent_state^next_state;

```

关键代码
状态机的第二段根据状态图里面的状态切换条件来产生次态

```
//////////状态机第一段（现态、次态转换）//////////
always @ (posedge clk_10MHz or negedge rst_n) begin
    if (!rst_n) begin
        persent_state=s0
    end
    else begin
        persent_state=next_state
    end
end
//////////状态机第二段实现状态转移判断//////////
always @(*) begin
    case(persent_state)
        s0:begin
            next_state = s1
        end
        s1:begin
            if (timer_1s==3'b011) begin
                next_state=s2
            end
            else
                next_state=s1
        end
        s2:begin
            if (timer_1s==3'b011) begin
                next_state=s3
            end
            else
                next_state=s2
        end
        s3:begin
            if (timer_1s==3'b011) begin
                next_state=s4
            end
            else
                next_state=s3
        end
        s4:begin
            if (timer_1s==3'b011) begin
                next_state=s5
            end
            else
                next_state=s4
        end
        s5:begin
            if (timer_1s==3'b011) begin
                next_state=s2
            end
            else
                next_state=s5
        end
        default : next_state=s0;
    endcase
end
```

如何使用状态机-Lab5:交通灯



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

红绿灯状态转换模块

关键代码
状态机的第
三段根据
timer_1s的
计数值输出
相应的动作

```
//////////状态机第三段(现态下的动作)/////////
always @ (posedge clk_10MHz or negedge rst_n) begin
    if (!rst_n) begin
        led_R_line <= 1'b1      ; 复位
        led_G_line <= 1'b1      ; 全亮
        led_Y_line <= 1'b1
        led_R_vertical <= 1'b1
        led_G_vertical <= 1'b1
        led_Y_vertical <= 1'b1
    end
    else begin
        case(present_state)
            s0:begin
                led_R_line <= 1'b1      ;
                led_G_line <= 1'b1      ; S0 全亮
                led_Y_line <= 1'b1
                led_R_vertical <= 1'b1
                led_G_vertical <= 1'b1
                led_Y_vertical <= 1'b1
            end
            s1:begin
                case(timer_1s)
                    s1'd0: begin
                        led_R_line <= 1'b0      ;
                        led_G_line <= 1'b0      ; S1 全灭
                        led_Y_line <= 1'b0
                        led_R_vertical <= 1'b0
                        led_G_vertical <= 1'b0
                        led_Y_vertical <= 1'b0
                    end
                    s1'd1: begin
                        led_R_line <= 1'b1      ; S1 全亮
                        led_G_line <= 1'b1
                        led_Y_line <= 1'b1
                        led_R_vertical <= 1'b1
                        led_G_vertical <= 1'b1
                        led_Y_vertical <= 1'b1
                    end
                endcase
            end
        endcase
    end
end
```

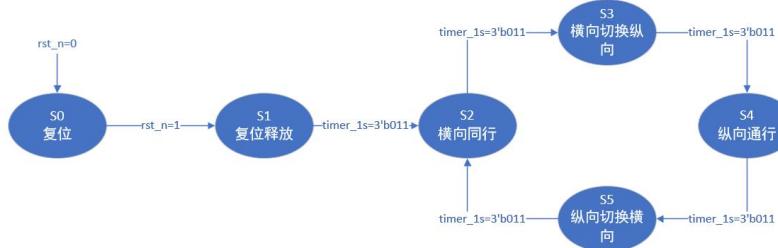
```
8'd2:begin
    led_R_line <= 1'b0      ; S1 全灭
    led_G_line <= 1'b0
    led_Y_line <= 1'b0
    led_R_vertical <= 1'b0
    led_G_vertical <= 1'b0
    led_Y_vertical <= 1'b0
end
default:begin
    led_R_line <= led_R_line ;
    led_G_line <= led_G_line ;
    led_Y_line <= led_Y_line ;
    led_R_vertical <= led_R_vertical;
    led_G_vertical <= led_G_vertical;
    led_Y_vertical <= led_Y_vertical;
end
endcase
end
s2:begin
    led_R_line <= 1'b0      ; S2 横向绿
    led_G_line <= 1'b1
    led_Y_line <= 1'b0      ; 纵向红
    led_R_vertical <= 1'b1
    led_G_vertical <= 1'b0
    led_Y_vertical <= 1'b0
end
s3:begin
    led_R_line <= 1'b0      ; S3 横向黄
    led_G_line <= 1'b0      ; 纵向红
    led_Y_line <= 1'b1
    led_R_vertical <= 1'b1
    led_G_vertical <= 1'b0
    led_Y_vertical <= 1'b0
end
end
endcase
end
endmodule
```

```
s4:begin
    led_R_line <= 1'b1      ; S4 横向红
    led_G_line <= 1'b0
    led_Y_line <= 1'b0      ; 纵向绿
    led_R_vertical <= 1'b0
    led_G_vertical <= 1'b1
    led_Y_vertical <= 1'b0
end
s5:begin
    led_R_line <= 1'b1      ; S5 横向红
    led_G_line <= 1'b0
    led_Y_line <= 1'b0      ; 纵向黄
    led_R_vertical <= 1'b0
    led_G_vertical <= 1'b0
    led_Y_vertical <= 1'b1
end
default:begin
    led_R_line <= led_R_line ;
    led_G_line <= led_G_line ;
    led_Y_line <= led_Y_line ;
    led_R_vertical <= led_R_vertical;
    led_G_vertical <= led_G_vertical;
    led_Y_vertical <= led_Y_vertical;
end
endcase
end
end
endmodule
```

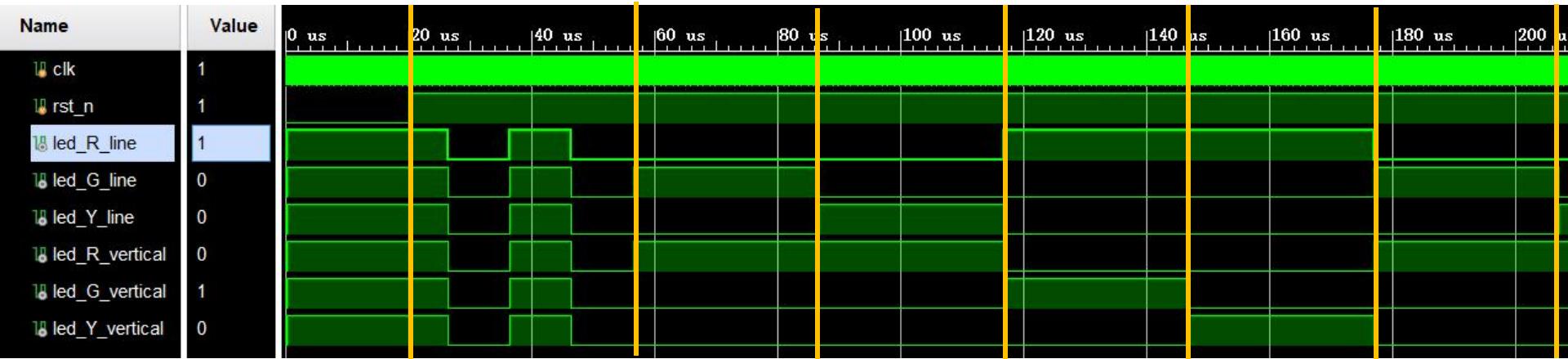
如何使用状态机-Lab5:交通灯



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



仿真结果



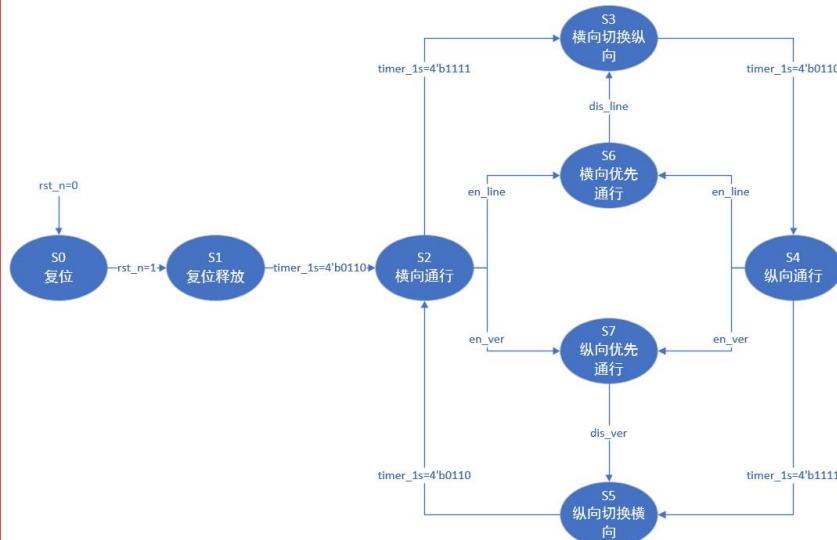
- S0 • S1 • S2 • S3 • S4 • S5 • S1
- 复位 • 复位释放 • 横向通行 • 横向切换纵向 • 纵向通行 • 纵向切换横向 • 横向通行

如何使用状态机-Lab5:交通灯



■ 效果2：修改红绿灯配时，增加优先通行功能

- [状态S0-复位]：复位状态下，横向及纵向所有灯全亮；
- [状态S1-复位释放后切换]：复位释放后，横向红灯闪烁3次（1Hz）->横向黄灯亮起（保持3秒），纵向保持红灯亮起；
- [状态S2-横向通行]：横向绿灯亮起（15秒），纵向保持红灯亮起；
 检测到横向优先通行，跳转到状态S6；检测到纵向优先通行，跳转到状态S7；
- [状态S3-横向切换纵向]：横向绿灯闪烁3秒（1Hz）->横向黄灯亮起（保持3秒），纵向保持红灯亮起；
- [状态S4-纵向通行]：纵向绿灯亮起（15秒），横向同时红灯亮起；
 检测到纵向优先通行，跳转到状态S7；检测到横向优先通行，跳转到状态S6；
- [状态S5-纵向切换横向]：纵向绿灯闪烁3秒（1Hz）->纵向黄灯亮起（保持3秒），横向保持红灯亮起；
- 返回状态S2，保持状态S2~状态S5间循环，复位键按下后回到状态S0；
- [状态S6-横向优先]：横向绿灯常亮，纵向红灯常亮；
 检测到退出横向优先通行，跳转到状态S3，否则保持状态S6；
- [状态S7-纵向优先]：横向红灯常亮，纵向绿灯常亮；
 检测到退出纵向优先通行，跳转到状态S5，否则保持状态S7；





**电路所有输入输出保持不变
仅需修改红绿灯状态转换模块
核心是状态图的设计（状态、状态切换条件、动作）**

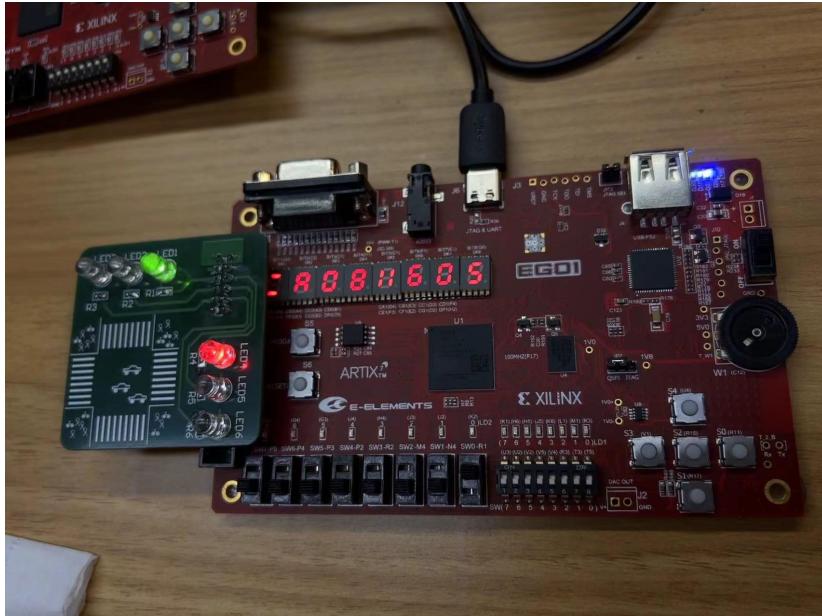
如何使用状态机-Lab5:交通灯



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

进阶版——

结合Lab4的数码管实验，增加倒计数功能





状态机在FPGA设计中具有重要作用

必须熟练掌握

必须熟练掌握

必须熟练掌握



THANKS



國科大杭州高等研究院
Hangzhou Institute for Advanced Study, UCAS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计

第八讲 (8.1 FPGA时序分析基础)

2025年4月22日

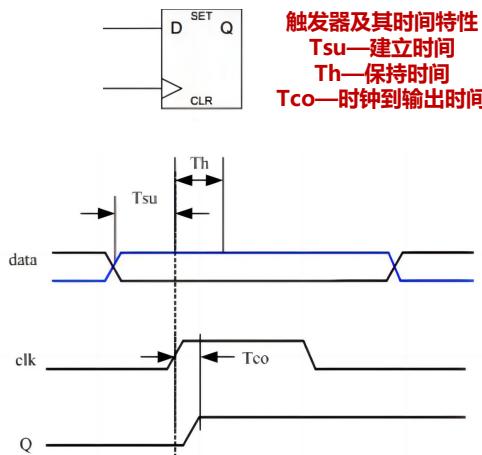
再谈时序逻辑与组合逻辑



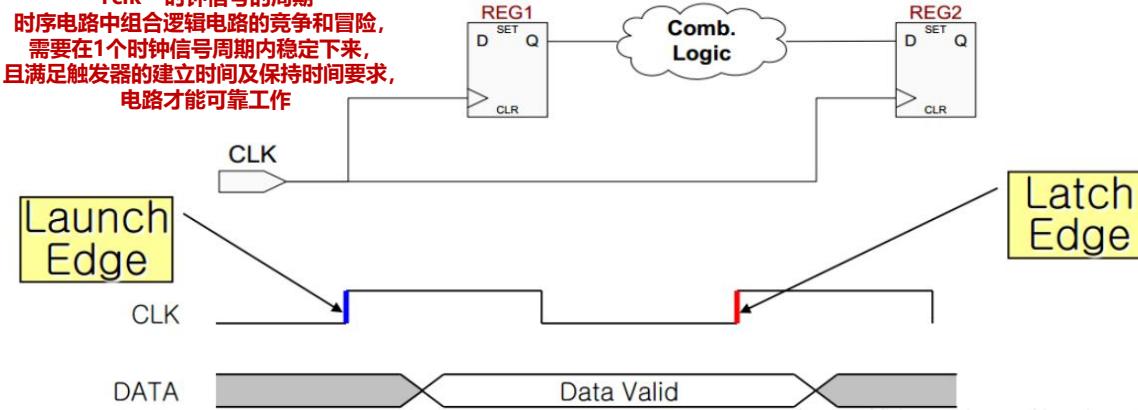
■ 时序逻辑电路

- 从电路功能上：FPGA设计中99.99%为时序逻辑电路，时序逻辑电路可以解决组合逻辑电路的竞争与冒险问题
- 时序逻辑电路中通常包含一部分的组合逻辑电路，这部分组合逻辑电路同样存在竞争和冒险问题
- 触发器的信号（D端）需要满足一定的时间特性才能可靠工作

时序逻辑电路如何才能可靠地工作？



T_{clk}—时钟信号的周期
时序电路中组合逻辑电路的竞争和冒险，
需要在1个时钟信号周期内稳定下来，
且满足触发器的建立时间及保持时间要求，
电路才能可靠工作



FPGA中时序的重要性



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

```
module Lab5_top(
    input sys_clk_in,
    input sys_rst_n,
    input [2:0] speed_sel,
    output [2:0] sel_display,
    output [7:0] seg7_cs,
    output [7:0] seg7_data0,
    output [7:0] seg7_data1
);

wire clk_10MHz;
wire rst_n;
wire rst;
wire [31:0] display_num;

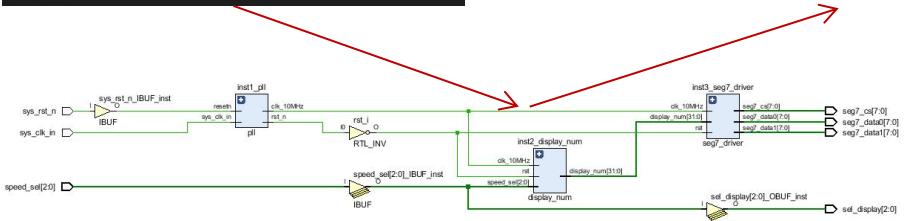
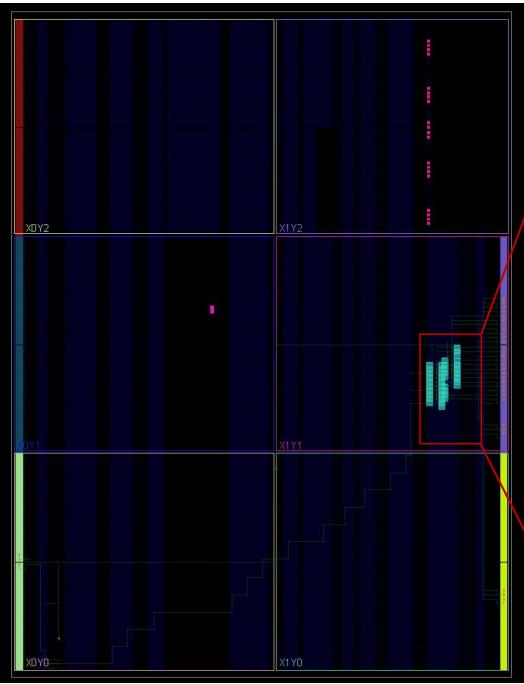
assign rst = ~rst_n;
assign sel_display = speed_sel;

// PLL
pll inst1_pll
(
    .clk_10MHz(clk_10MHz),
    .resetn(sys_rst_n),
    .rst_n(rst_n),
    .sys_clk_in(sys_clk_in)
);

display_num inst2_display_num
(
    .clk_10MHz(clk_10MHz),
    .rst(rst),
    .speed_sel(speed_sel),
    .display_num(display_num)
);

seg7_driver inst3_seg7_driver
(
    .clk_10MHz(clk_10MHz),
    .rst(rst),
    .display_num(display_num),
    .seg7_cs(seg7_cs),
    .seg7_data0(seg7_data0),
    .seg7_data1(seg7_data1)
);

endmodule
```



- 使用verilog进行电路描述时没有时间特性参数
- 功能仿真不考虑电路时间特性
- 必要时需要告知编译器电路中的某些时间特性

FPGA中时序的重要性



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ FPGA的时间延迟特性

- FPGA的内部结构：LUT、触发器、嵌入式模块（RAM、乘法器等）等具有时间延迟特性
- 走线延迟：信号经过走线必定有时间延迟，尤其当FPGA内部资源紧张的情况下，信号可能会“绕路”，走更长的路径，产生更大的时间延迟
- 门延迟：组合逻辑电路经过多个LUT也会产生延迟
- FPGA内部的延迟不可忽略，通常达到ns级以上
- 时序如果不能满足要求，则电路不一定能可靠甚至不能正常工作

■ 时序分析

- 时序分析本质是一种时序检查，目的是检查设计中所有的触发器是否能够可靠工作
- 检查触发器数据输入端口的时间特性是否满足建立时间（Setup）和保持时间要求（Hold）

■ 时序约束

- 告知编译器，明确电路的时间特性，编译器在布局布线时通过算法尽力满足设计的需求

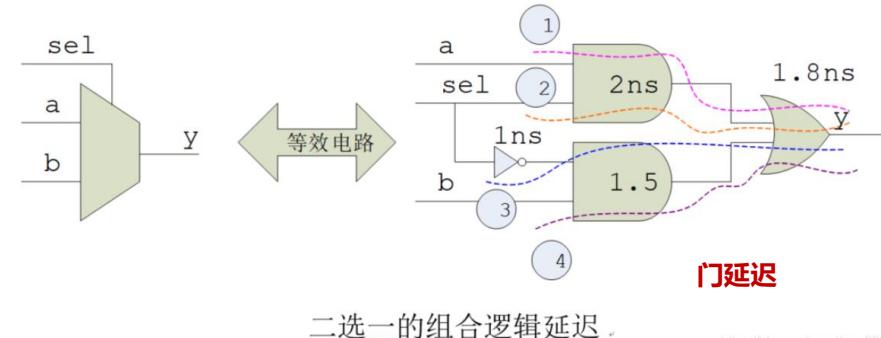
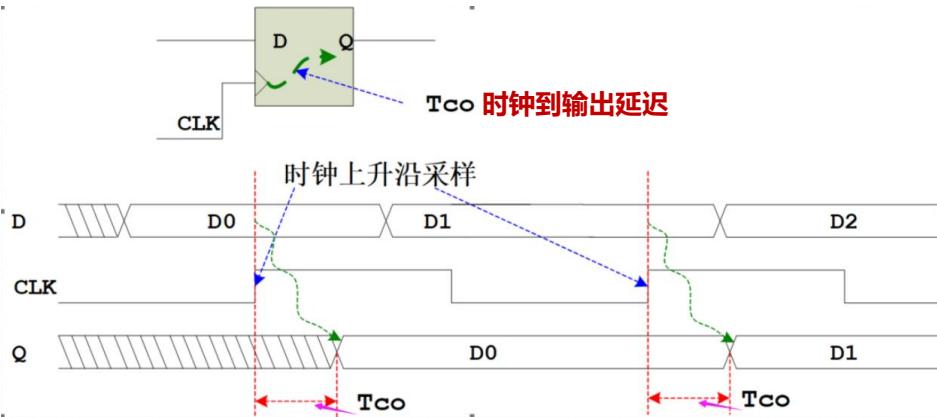
电路受到器件温度、内核电压影响，时间特性会改变，不满足建立时间及保持时间要求，电路不一定能可靠地工作！

数字电路的延迟



■ 走线延迟、门延迟、时钟到输出延迟

- 电气走线一定会引入延迟，称之为走线延迟
- 对于组合逻辑电路，电信号经过逻辑门电路时，同样会造成延迟，称之为门延迟
- 对于时序逻辑电路，时钟触发到数据输出存在一定的延迟，称之为时钟到输出延迟



$$V = C/E_r^{0.5}$$

$$1\text{inch} = 1000\text{mil} = 25.4\text{mm}$$

走线延迟

介质	延迟(ps/in)	介电常数
空气 (无线电波)	85	1.0
同轴电缆 (75%速度)	113	1.8
同轴电缆 (66%速度)	129	2.3
FR-4 PCB (表层微带线)	140~180	2.8~4.5
FR-4 PCB (内层带状线)	180	4.5
氧化铝PCB (内层带状线)	240~270	8~10

二选一的组合逻辑延迟

数字电路的延迟



XILINX®

Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics

CLB Switching Characteristics

Table 27: CLB Switching Characteristics

Symbol	Description	Speed Grade						Units
		1.0V			0.95V	0.9V		
		-3	-2/-2LE	-1	-1Q/-1M	-1LI	-2LE	
Combinatorial Delays								
T_{ILO}	An - Dn LUT address to A	0.10	0.11	0.13	0.13	0.13	0.15	ns, Max
T_{ILO_2}	An - Dn LUT address to AMUX/CMUX	0.27	0.30	0.36	0.36	0.36	0.41	ns, Max
T_{ILO_3}	An - Dn LUT address to BMUX_A	0.42	0.46	0.55	0.55	0.55	0.65	ns, Max
T_{ITO}	An - Dn inputs to A - D Q outputs	0.94	1.05	1.27	1.27	1.27	1.51	ns, Max
T_{AXA}	AX inputs to AMUX output	0.62	0.69	0.84	0.84	0.84	1.01	ns, Max
T_{AXB}	AX inputs to BMUX output	0.58	0.66	0.83	0.83	0.83	0.98	ns, Max
T_{AXC}	AX inputs to CMUX output	0.60	0.68	0.82	0.82	0.82	0.98	ns, Max
T_{AXD}	AX inputs to DMUX output	0.68	0.75	0.90	0.90	0.90	1.08	ns, Max
T_{BXB}	BX inputs to BMUX output	0.51	0.57	0.69	0.69	0.69	0.82	ns, Max
T_{BxD}	BX inputs to DMUX output	0.62	0.69	0.82	0.82	0.82	0.99	ns, Max
T_{CXC}	CX inputs to CMUX output	0.42	0.48	0.58	0.58	0.58	0.69	ns, Max
T_{CXD}	CX inputs to DMUX output	0.53	0.59	0.71	0.71	0.71	0.86	ns, Max
T_{DXD}	DX inputs to DMUX output	0.52	0.58	0.70	0.70	0.70	0.84	ns, Max
Sequential Delays								
T_{CKO}	Clock to AQ - DQ outputs	0.40	0.44	0.53	0.53	0.53	0.62	ns, Max
T_{SHCKO}	Clock to AMUX - DMUX outputs	0.47	0.53	0.66	0.66	0.66	0.73	ns, Max
Setup and Hold Times of CLB Flip-Flops Before/After Clock CLK								
T_{AS}/T_{AH}	$A_N - D_N$ input to CLK on A - D flip-flops	0.07/0.12	0.09/0.14	0.11/0.18	0.11/0.28	0.11/0.18	0.11/0.22	ns, Min
$T_{CKD'}/T_{CKD}$	$A_Y - D_X$ input to CLK on A - D flip-flops	0.06/0.19	0.07/0.21	0.09/0.26	0.09/0.35	0.09/0.26	0.09/0.33	ns, Min
	$A_X - D_X$ input through MUXs and/or carry logic to CLK on A - D flip-flops	0.59/0.08	0.66/0.09	0.81/0.11	0.81/0.20	0.81/0.11	0.97/0.15	ns, Min

Artix-7系列FPGA的时间特性参数

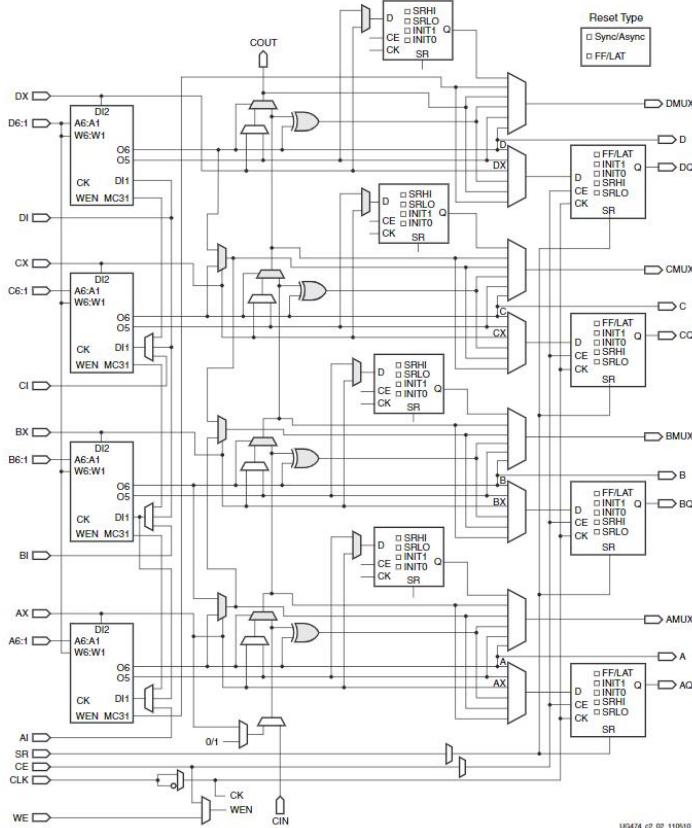


Figure 2-3: Diagram of SLICEM

UG474_c2_02_110510

FPGA时序分析基础



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 建立时间 (Tsu)

- 采样时刻（时钟上升/下降沿）之前，数据需要稳定的最短时间，为建立时间

■ 保持时间 (Th)

- 采样时刻（时钟上升/下降沿）之后，数据需要维持的最短时间，为保持时间

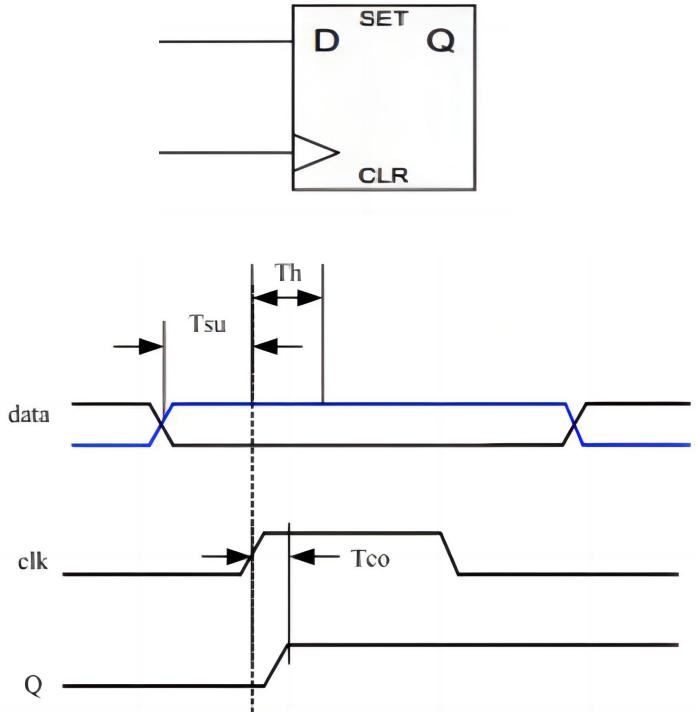
■ 亚稳态

- 指触发器在某一段时间内不能达到一个确定的状态
- 触发器的输出端Q在较长时间内处于振荡状态，不等于输入端D
- 振荡状态的时间称作决断时间 (resolution time)
- 亚稳态出现的主要原因是由于触发器无法满足建立时间或者保持时间要求

■ 亚稳态的传播

- 亚稳态震荡会导致传播到下一级，引起更大范围的故障
- 改善方法：

优化时序[降低时钟频率，用更快的触发器，两级同步，改善时钟质量]



ALTERA

White Paper

Understanding Metastability in FPGAs



■ 触发沿 (Launch Edge)

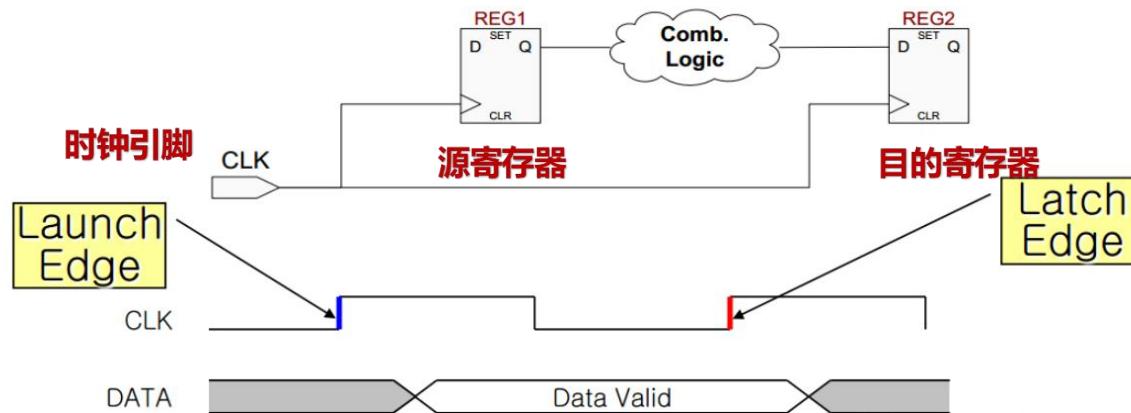
- 源寄存器的时钟触发边沿

■ 锁存沿 (Latch Edge)

- 目的寄存器的时钟锁存边沿

■ 若触发沿定义为T=0ns，锁存沿则为延迟1个时钟周期(Tclk)

① 时序逻辑电路的简化模型



FPGA时序分析基础



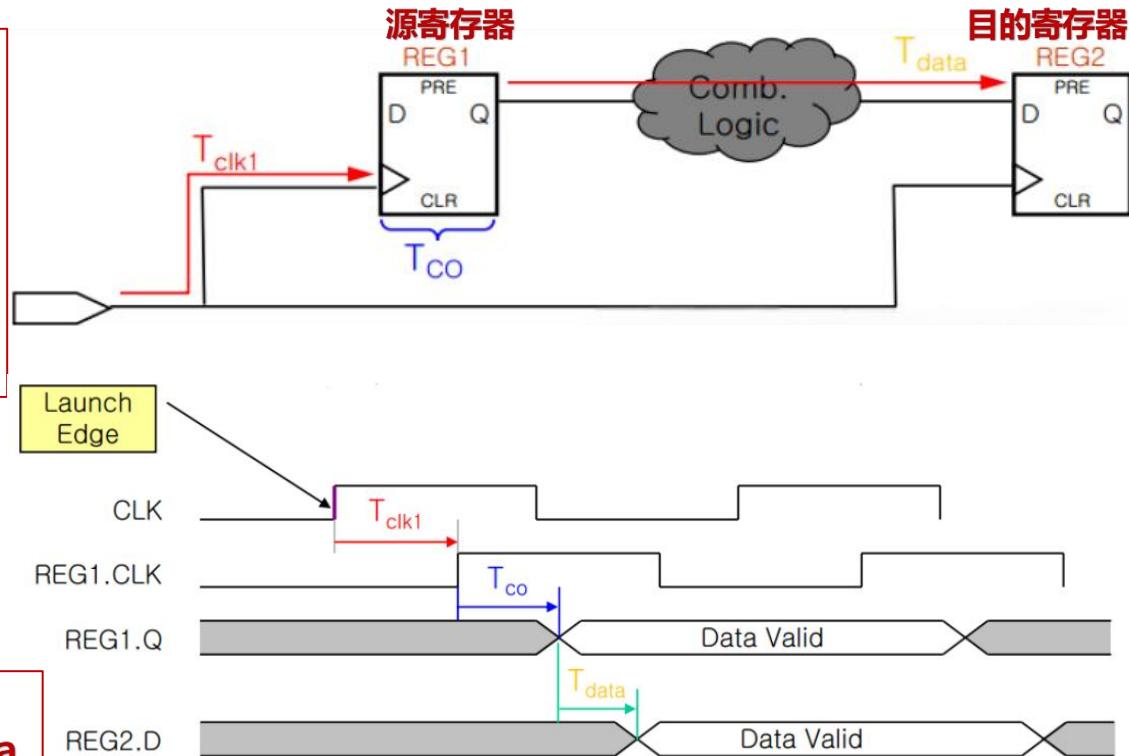
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 数据到达时间(Data arrival time)

- 数据到达目的寄存器的时间
- T_{clk1} : 触发沿到源寄存器的时钟延迟
- T_{co} : 源寄存器时钟到输出延迟
- T_{data} : REG1的Q端数据到达REG2的D端所消耗时间 (信号路径存在组合逻辑及走线延迟)

② 数据到达时间

Data arrival time = Launch edge +
 T_{clk1} + T_{co} + T_{data}



FPGA时序分析基础

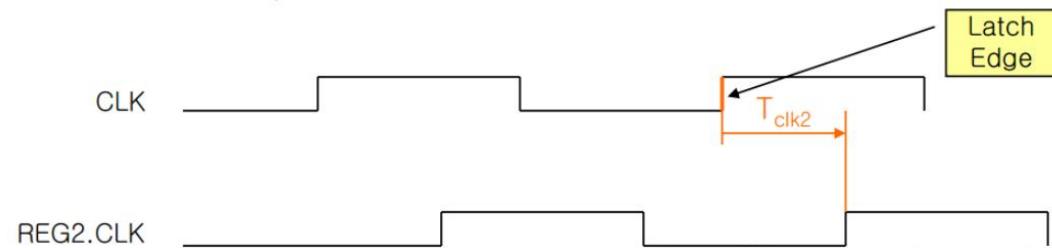
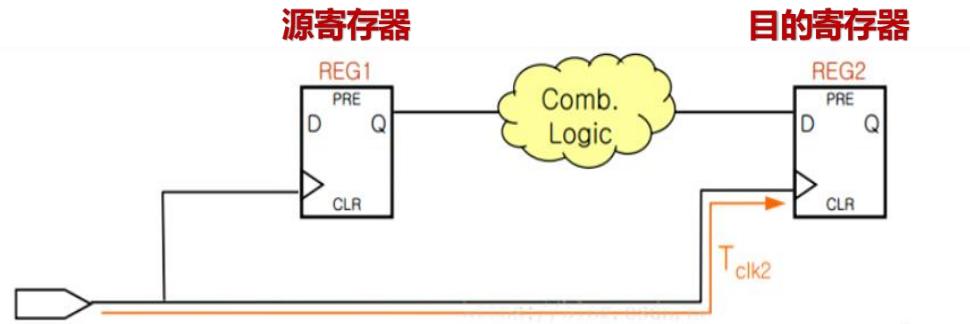


國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 时钟到达时间(Clock arrival time)

- 时钟到达目的寄存器的时间
- T_{clk2} :锁存沿到目的寄存器的时钟延迟

③ 时钟到达时间



$$\text{Clock arrival time} = \text{Latch edge} + T_{clk2}$$

FPGA时序分析基础



■ 数据锁存到目的寄存器的要求时间(Data required time)

- 能够被目的寄存器稳定锁存的要求时间
- 包括建立时间要求及保持时间要求

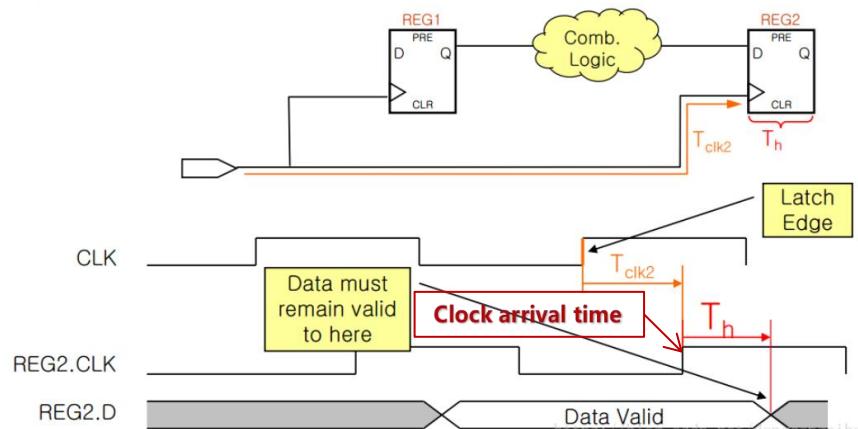
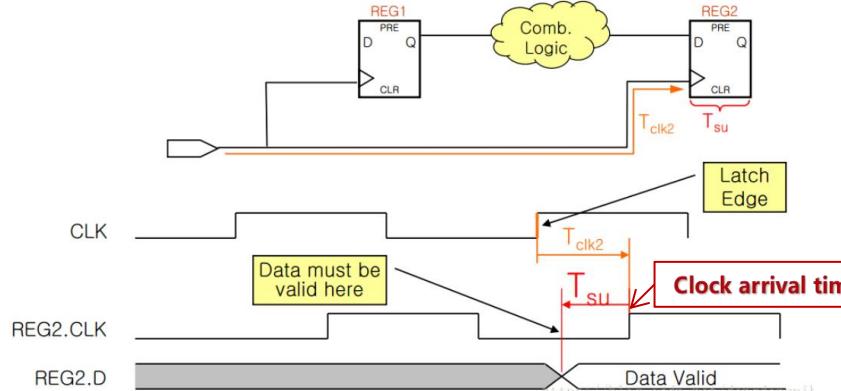
④ 时序要求

建立时间要求:

$$\text{Data required time}(T_{SU}) = \text{Clock arrival time} - T_{SU}$$

保持时间要求:

$$\text{Data required time}(T_h) = \text{Clock arrival time} + T_h$$



FPGA时序分析基础



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 建立时间裕量 (Setup slack) :

- 满足建立时间时序要求的时间余量
- 确保源端的数据及时到达，以满足目的寄存器锁定要求

⑤ 时间裕量——建立时间

Setup slack = Data required time(Tsu) - Data arrival time(current)

其中：Data required time(Tsu)= Clock arrival time - Tsu

Clock arrival time = Latch edge + Tclk2

Data arrival time(current) = Launch edge + Tclk1 + Tco + Tdata

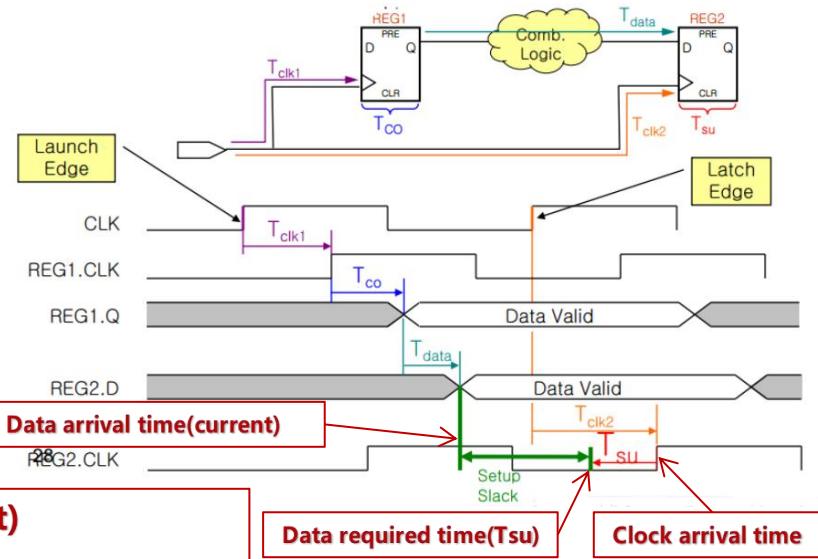
Setup slack = Latch edge + Tclk2 - Tsu - (Launch edge + Tclk1 + Tco + Tdata)

= Latch edge + Tclk2 - Tsu - Launch edge - Tclk1 - Tco - Tdata

FPGA中使用全局时钟网络，Tclk1 = Tclk2

不考虑时钟抖动，Latch edge - Launch edge = Tclk (时钟周期)

Setup slack = Tclk - Tsu - Tco - Tdata



Tco,Tsu为器件固有特性

减小Tdata可以增大建立时间裕量

即电路主频可以跑得更快

FPGA时序分析基础



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 保持时间裕量 (Hold slack) :

- 满足保持时间时序要求的时间余量
- 确保新的Launch Edge引起的数据变化，满足目的寄存器保持时间要求

⑤ 时间裕量——保持时间

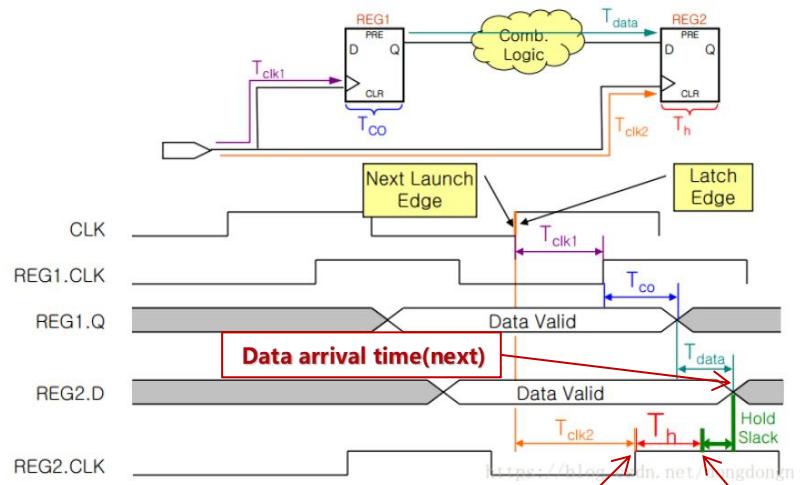
Hold slack = Data arrival time(next) - Data required time(Th)

其中: **Data arrival time(next) = Latch edge + Tclk1 + Tco + Tdata**
Data required time(Th) = Clock arrival time + Th
= Latch edge + Tclk2 + Th

Hold slack = Latch edge + Tclk1 + Tco + Tdata - (Latch edge + Tclk2 + Th)
= Tclk1 + Tco + Tdata - Tclk2 - Th

FPGA中使用全局时钟网络, **Tclk1 = Tclk2**

Hold slack = Tco + Tdata - Th



Tco, Th为器件固有特性

增大Tdata可以增大保持时间裕量

时序分析及时序约束的目标



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 时序裕量 (Slack) 不能为负，且尽可能大

- 裕量 (Slack) 是实际所用时间和设计所需时间的差值，表示设计是否满足时序的一个量化指标
- 正的Slack表示满足时序（时序的裕量）
- 负的Slack表示不满足时序（时序的欠缺量）
- Slack分为建立时间裕量和保持时间裕量
- 通过对电路设计进行时间特性约束来满足建立时间裕量和保持时间裕量

时序分析分类

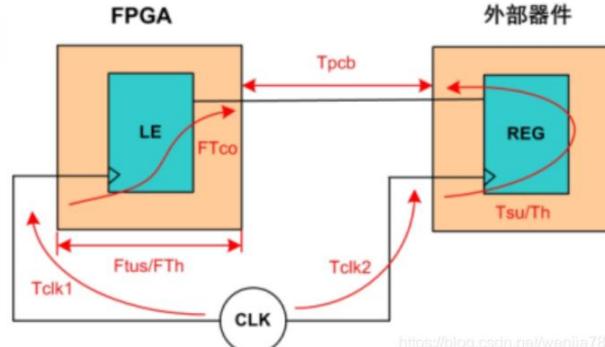
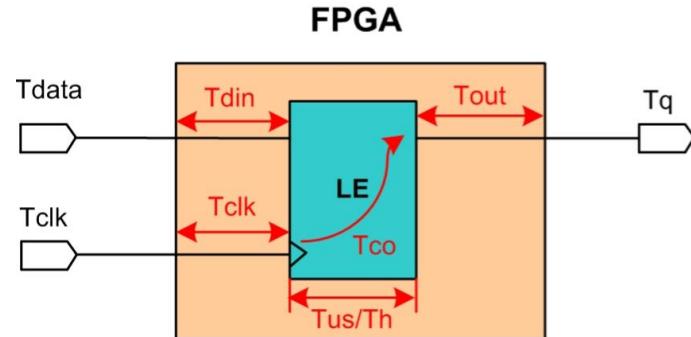


■ 芯片内部时序分析（寄存器与寄存器之间）

- 静态时序分析：根据电路的已知参数，分析时钟和数据的关系，不需要电路模拟运行
- 动态时序分析（时序仿真）：让电路模拟运行，考虑所有的延迟，根据仿真波形进行时序和功能分析

■ IO时序分析

- T_{din} 为从FPGA的IO口到FPGA内部寄存器数据输入端的延时
- T_{clk} 为从FPGA的IO口到FPGA内部寄存器时钟输入端的延时
- $T_{su/Th}$ 为FPGA内部寄存器的建立时间和保持时间
- T_{co} 为FPGA内部寄存器传输时间
- T_{out} 为从FPGA寄存器输出到IO口输出的延时
- 板级的IO时序分析（板级时钟延迟 T_{clk1} 、 T_{clk2} 、板级走线延迟 T_{pcb} 等）



时序约束



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

The screenshot shows the Vivado Design Suite interface with the following details:

- Top Bar:** File, Edit, Flow, Tools, Reports, Window, Layout, View, Help, Quick Access.
- Title Bar:** IMPLEMENTED DESIGN - xc7a35tcsg324-1, write_bitstream Complete ✓, Default Layout.
- Left Sidebar (Flow Navigator):**
 - RTL ANALYSIS: Open Elaborated Design
 - SYNTHESIS: Run Synthesis, Open Synthesized Design
 - IMPLEMENTATION:** Run Implementation, **Open Implemented Design** (highlighted), Constraints Wizard, Edit Timing Constraints (highlighted).
 - Report Timing Summary, Report Clock Networks, Report Clock Interaction, Report Methodology, Report DRC, Report Noise, Report Utilization, Report Power, Schematic, Generate Bitstream.
- Middle Area:**
 - Project Summary:** Lab3_seg7_top, Nets (107), Leaf Cells (25), dbg_hub (dbg_hub), inst1 (clk_wiz_0), inst2 (Lab3_seg7_driver), inst3 (ila_0).
 - Timing Constraints:** Rename Auto-Derived Clock, Position, Clock Name, Source/Master Pin, Master Clock, Source Objects, Source File, Scoped Cell, Current Object.
 - Properties:** Select an object to see properties.
- Bottom Area:** Tcl Console, Messages, Log, Reports, Design Runs, DRC, Methodology, Power, **Timing** (highlighted), Timer Settings, General Information, Timer Settings, Design Timing Summary, Clock Summary (4), Check Timing (22), Intra-Clock Paths, Inter-Clock Paths, Timing Summary - impl_1 (saved).

**Vivado时序分析工具
可以进行时序约束**



■ Vivado时序约束

- **主时钟 (Primary Clock) 约束**, 通常为外部主时钟
- **衍生时钟 (Generated Clocks) 约束**, 时钟模块或逻辑产生的时钟
- **时钟组约束**
- **时钟延迟约束**
- **时钟抖动约束**
- **时钟附加不确定性约束**
- **IO输入延迟约束**
- **IO输出延迟约束**

Vivado Design Suite User Guide

Using Constraints

UG903 (v2024.2) December 20, 2024

See all versions of this document

AMD Adaptive Computing is creating an environment where everyone feels included. Non-inclusive language is included. To that end, we're removing non-inclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You will see limited examples of non-inclusive language in our older products as we work to move these changes and align with evolving industry standards. Follow this [link](#) for more information.



时序报告分析



时序分析报告

Summary	
Name	Path 161
Slack	-0.852ns 余量 (为负说明时序违例)
Source	i_DPLL_phase_A/i_Multi_sin/U0/i_mult/gDSP.gDSP_only.iDSP/inferred_dsp.Pdelay/d1.dout_i_reg/CLK (rising edge-triggered cell DSP48E1 clocked by adc_userclk_buf {rise@0.000ns fall@2.000ns})
Destination	i_DPLL_phase_A/IIR/Zero/Xin_reg/reg[0][9]/D (rising edge-triggered cell FDCE clocked by clk_out1_clk_wiz_1_1 {rise@0.000ns fall@50.000ns period=100.000ns})
Path Group	clk_out1_clk_wiz_1_1
Path Type	Setup (Max at Slow Process Corner) 建立时间
Requirement	4.000ns (clk_out1_clk_wiz_1_1 rise@100.000ns - adc_userclk_buf rise@96.000ns)
Data Path Delay	7.824ns (logic 0.348ns (4.448%) route 7.476ns (95.552%))
Logic Levels	0
Clock Path Skew	3.386ns
Clock Uncertainty	0.414ns
Clock Dom...Crossing	Inter clock paths are considered valid unless explicitly excluded by timing constraints such as set_clock_groups or set_false_path.

Design Timing Summary

General Information	Setup	Hold	Pulse Width
Timer Settings	Worst Negative Slack (WNS): -1.034 ns	Worst Hold Slack (WHS): 0.045 ns	Worst Pulse Width Slack (WPWS): 1.250 ns
Design Timing Summary	Total Negative Slack (TNS): -8.271 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Clock Summary (1)	Number of Failing Endpoints: 8	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
> Check Timing (1)	Total Number of Endpoints: 3470	Total Number of Endpoints: 3470	Total Number of Endpoints: 1480
> Intra-Clock Paths	Timing constraints are not met.		
Inter-Clock Paths			
> Other Path Groups			
> User Ignored Paths			
> Unconstrained Paths			



如何提高系统可运行的时钟速率?

■ 建立时间裕量: $T_{clk} - T_{co} - T_{data} - Tsu \geq 0$

- 满足不等式的条件下, 减小时钟周期 T_{clk} 既提高时钟速率

$$T_{clk(min)} = T_{co} + T_{data} + Tsu$$

- T_{co} 、 Tsu 均为系统内部参数无法修改

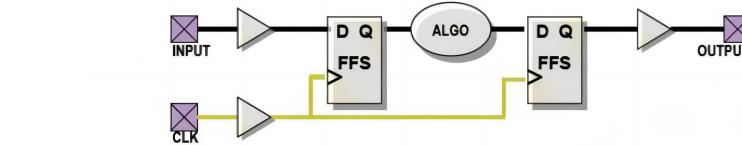
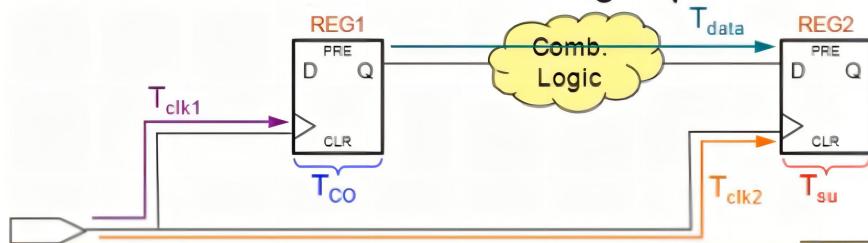
■ 减小源寄存器到目的寄存器之间的 T_{data}

- 通过约束寄存器到寄存器之间的组合逻辑时延, 但这种方式不能大幅度提高时钟速率, 限制组合逻辑时延不利于系统布局布线

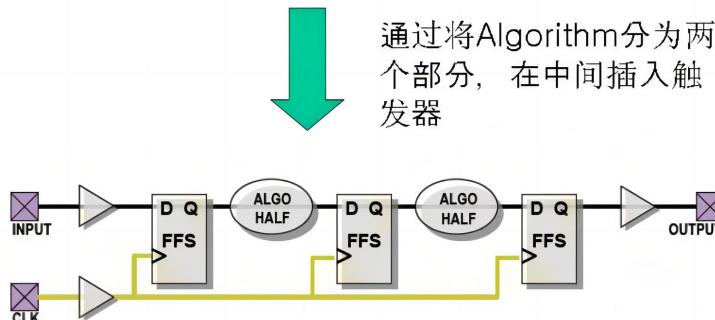
■ 降低逻辑判断的复杂性

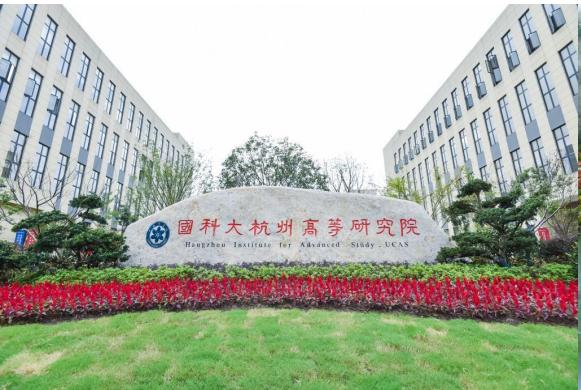
■ 采用流水线设计

- 流水线设计思想: 减少大规模组合逻辑的使用, 将大规模组合逻辑拆分为多个小规模组合逻辑, 减小组合逻辑复杂度, 放入多级寄存器之间, 提高系统运行时钟的工作频率。(面积与速度互换)



通过将Algorithm分为两个部分, 在中间插入触发器





THANKS



國科大杭州高等研究院
Hangzhou Institute for Advanced Study, UCAS



國科大杭州高
等研究
院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第八讲 (8.2 Lab6数码管实验2)

2025年4月22日

Lab6数码管实验2



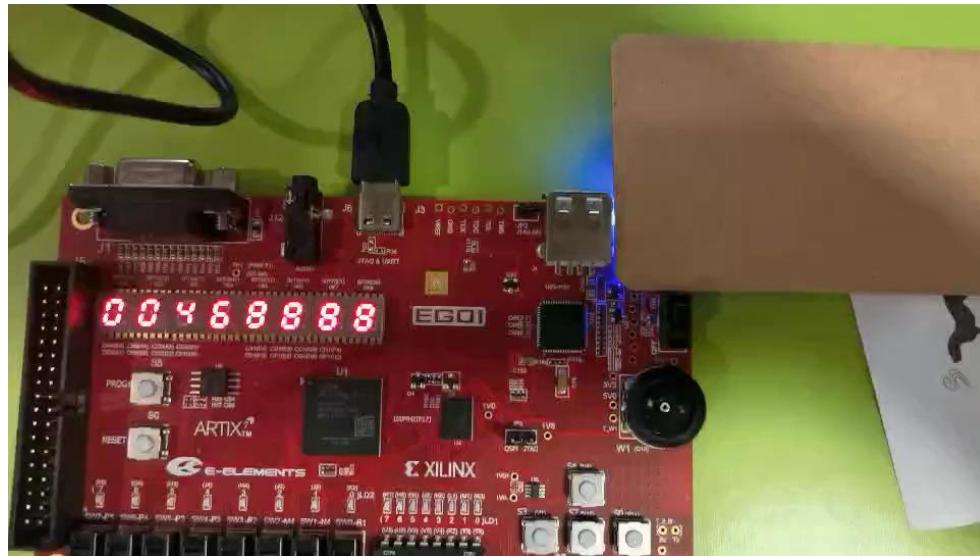
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 实验内容

- 学会数码管的动态显示控制
- 学会BCD计数器的设计
- 效果1：
利用8颗数码管完成信息显示
显示二进制计数器数值
- 效果2：
利用8颗数码管完成信息显示
显示BCD计数器数值

■ 实验目的

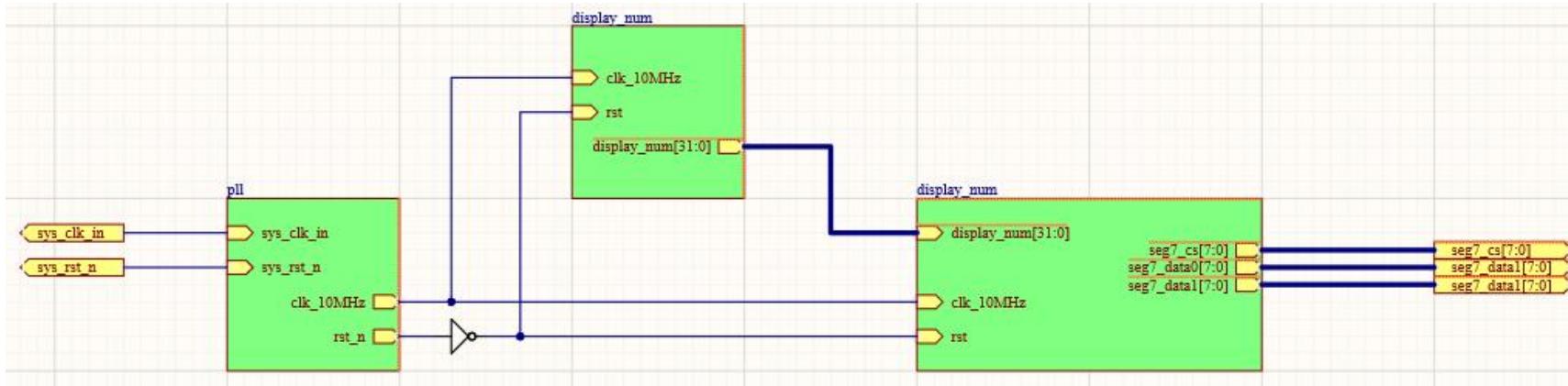
- 理解数码管动态刷新显示控制的基本方法
- 强化层次化结构设计思想及能力



Lab6数码管实验2



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



■ 系统框图

- 电路包括三个模块：pll模块实现时钟管理，display_num模块产生需要显示的数据，seg7_driver模块完成数码管驱动控制
- pll模块：通过IP核实现，100MHz时钟输入，全局复位引脚输入，生成10MHz信号，频率锁定信号用于后续模块复位
- display_num模块：10MHz时钟输入，频率锁定信号作为复位信号，输出需要显示的数据（每位数码管数据用4位表示）
- seg7_driver模块：10MHz时钟输入，频率锁定信号作为复位信号，通过seg7_cs,seg7_data控制数码管点亮

Lab6数码管实验2



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

输入输出端口表

序号	信号名称	信号类型	输入/输出	信号描述	FPGA引脚	说明
1	sys_clk_in	1bit	输入	系统时钟输入	P17	100MHz输入
2	sys_RST_n	1bit	输入	系统复位输入	P15	低电平复位
3	seg7_cs[7:0]	8bit	输出	数码管位选控制信号输出	G6/E1/F1/G1 H1/C1/C2/G2	高电平选通
4	seg7_data0[7:0]	8bit	输出	数码管字段控制信号输出	D5/B2/B3/A1 B1/A3/A4/B4	高电平有效
5	seg7_data1[7:0]	8bit	输出	数码管字段控制信号输出	H2/D2/E2/F3 F4/D3/E3/D4	高电平有效

Lab6数码管实验2



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

模块输入输出端口表——锁相环模块 (pll)

序号	信号名称	信号类型	输入/输出	信号描述	说明
1	sys_clk_in	1bit	输入	系统时钟输入	100MHz输入
2	sys_RST_n	1bit	输入	系统复位输入	低电平复位
3	clk_10MHz	1bit	输出	系统主时钟	10MHz时钟输出
4	rst_n	1bit	输出	锁相环时钟锁定输出	高电平锁定

模块输入输出端口表——显示数据产生模块 (display_num)

序号	信号名称	信号类型	输入/输出	信号描述	说明
1	clk_10MHz	1bit	输入	系统主时钟	10MHz时钟输入
2	rst	1bit	输入	系统主复位	高电平复位
3	display_num	32bit	输出	显示数据输出	32bit, 每颗数码管使用4bit



模块输入输出端口表——数码管驱动模块 (seg7_driver)

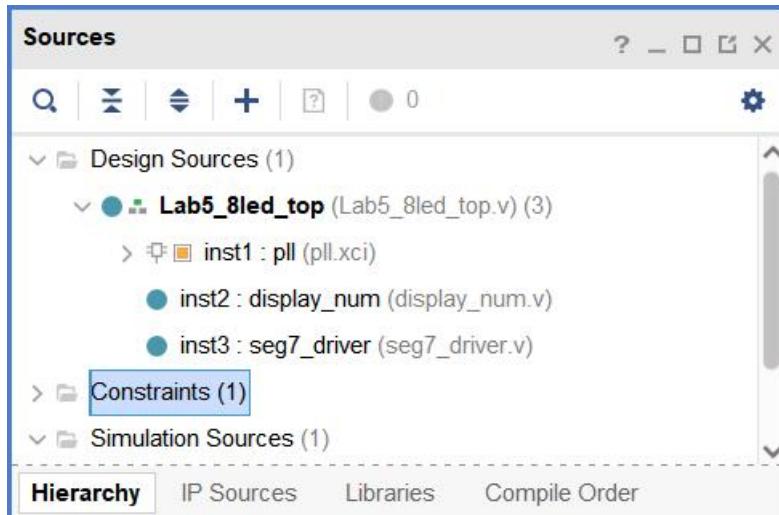
序号	信号名称	信号类型	输入/输出	信号描述	说明
1	clk_10MHz	1bit	输入	系统主时钟	10MHz时钟输入
2	rst	1bit	输入	系统主复位	高电平复位
3	display_num	32bit	输入	显示数据输入	32bit, 每颗数码管使用4bit
4	seg7_cs[7:0]	8bit	输出	数码管位选控制信号输出	高电平选通
5	seg7_data0[7:0]	8bit	输出	数码管字段控制信号输出	高电平有效
6	seg7_data1[7:0]	8bit	输出	数码管字段控制信号输出	高电平有效

Lab6数码管实验2



■ 实验步骤

- 建立工程
- 添加顶层设计文件, Lab6_8led_top.v
- 通过IP catalog生成时钟模块pll.xci(inst1)
- 设计计数模块display_num.v(inst2)
- 设计数码管驱动模块seg7_driver.v(inst3)
- 在顶层文件完成模块例化及互联
- 添加引脚约束文件
- 设计仿真激励文件
- 完成功能仿真
- 综合
- 实现
- 下载调试





关键代码

Lab6数码管实验2



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

```
// 定义数码管刷新周期计数器
reg [31:0] cnt_refresh;
// 定义当前正常刷新的数码管编号
reg [3:0] led_num = 4'd0;

// 刷新计数间隔设置为1ms, 1ms/100ns = 10_000
parameter REFRESH_TIME = 10_000;

// 定义当前显示数据
reg [3:0] current_display_num;

// 产生数码管刷新显示计数器, 每1ms切换一位
always @(posedge clk_10MHz or posedge rst) begin
    if(rst) begin
        cnt_refresh <= 32'd0;
        led_num <= 3'd0;
    end
    else if(cnt_refresh == (REFRESH_TIME-1)) begin
        cnt_refresh <= 32'd0;
        if(led_num == 4'd7) begin
            led_num <= 4'd0;
        end
        else begin      生成数码管刷新计数
            led_num <= led_num + 1'b1;
        end
    end
    else begin
        cnt_refresh <= cnt_refresh + 1'b1;
    end
end
```

```
//根据数码管刷新显示计数器, 选择输出当前的字段与位选信号
always @ (posedge clk_10MHz or posedge rst) begin
    if(rst) begin
        current_display_num <= NUM0;
        seg7_cs <= CSN;
    end
    else begin
        case(led_num)  根据数码管刷新计数值
            4'd0: begin
                current_display_num <= display_num[3:0];
                seg7_cs = CS0;
            end
            4'd1: begin
                current_display_num <= display_num[7:4];
                seg7_cs <= CS1;
            end
            4'd2: begin
                current_display_num <= display_num[11:8];
                seg7_cs <= CS2;
            end
            4'd3: begin
                current_display_num <= display_num[15:12];
                seg7_cs <= CS3;
            end
            4'd4: begin
                current_display_num <= display_num[19:16];
                seg7_cs <= CS4;
            end
            4'd5: begin
                current_display_num <= display_num[23:20];
                seg7_cs <= CS5;
            end
            4'd6: begin
                current_display_num <= display_num[27:24];
                seg7_cs <= CS6;
            end
            4'd7: begin
                current_display_num <= display_num[31:28];
                seg7_cs <= CS7;
            end
            default: begin
                current_display_num <= NUM0;
                seg7_cs <= CSN;
            end
        endcase
    end
end
```

数码管驱动模块(seg7_driver.v)
通过字库索引输出数据

```
//根据数码管显示数据进行字库索引
always @ (posedge clk_10MHz or posedge rst) begin
    if(rst) begin
        seg7_data0 <= NUM0;
    end
    else begin      字库索引输出
        case(current_display_num)
            4'h0: seg7_data0 <= NUM0;
            4'h1: seg7_data0 <= NUM1;
            4'h2: seg7_data0 <= NUM2;
            4'h3: seg7_data0 <= NUM3;
            4'h4: seg7_data0 <= NUM4;
            4'h5: seg7_data0 <= NUM5;
            4'h6: seg7_data0 <= NUM6;
            4'h7: seg7_data0 <= NUM7;
            4'h8: seg7_data0 <= NUM8;
            4'h9: seg7_data0 <= NUM9;
            4'ha: seg7_data0 <= NUMA;
            4'hb: seg7_data0 <= NUMB;
            4'hc: seg7_data0 <= NUMC;
            4'hd: seg7_data0 <= NUMD;
            4'he: seg7_data0 <= NUME;
            4'hf: seg7_data0 <= NUMF;
            default: seg7_data0 <= NUM0;
        endcase
    end
    seg7_data1 <= seg7_data0;
end
```

Lab6数码管实验2



國科大杭州高夢研究院
Hangzhou Institute for Advanced Study, UCAS

■效果2

- 8颗数码管动态显示
- BCD计数
- 三个按键控制计数速度

001: 1ms

010: 100ms

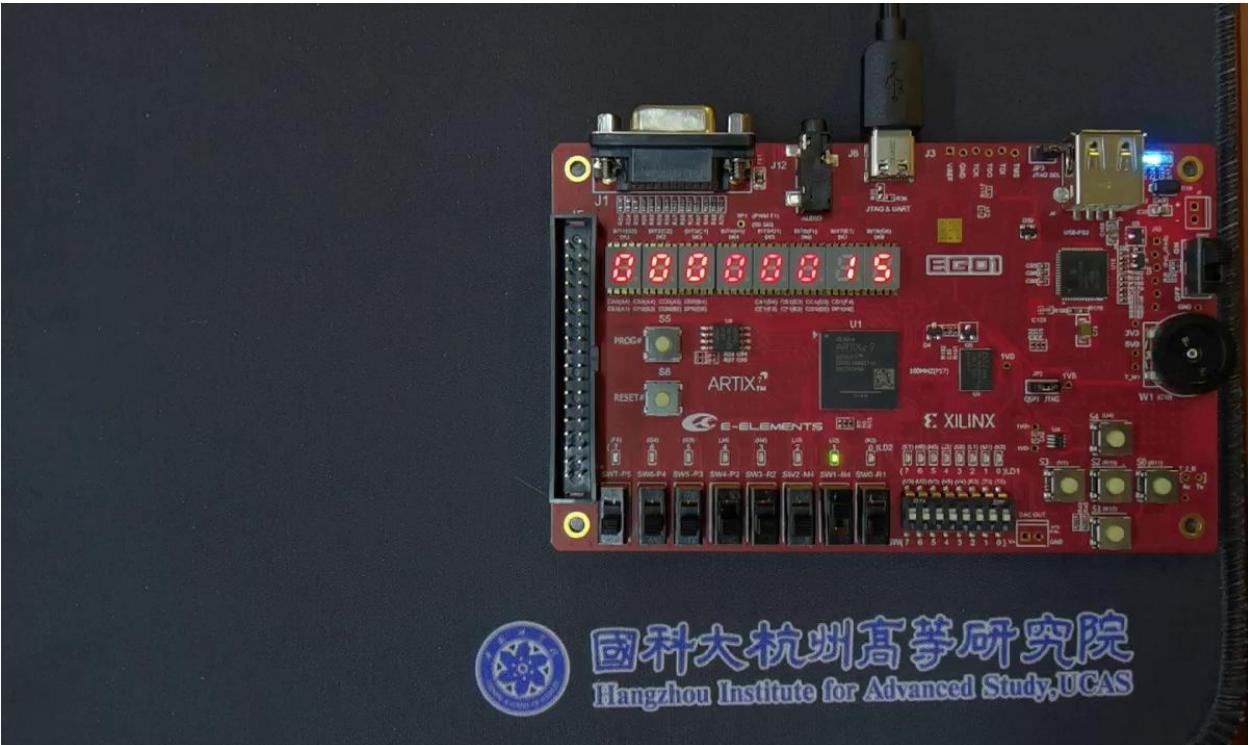
100: 1s

- 三颗LED显示当前计数速度

001: 1ms

010: 100ms

100: 1s



Lab6数码管实验2



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

输入输出端口表

序号	信号名称	信号类型	输入/输出	信号描述	FPGA引脚	说明
1	sys_clk_in	1bit	输入	系统时钟输入	P17	100MHz输入
2	sys_RST_n	1bit	输入	系统复位输入	P15	低电平复位
3	seg7_cs[7:0]	8bit	输出	数码管位选控制信号输出	G6/E1/F1/G1 H1/C1/C2/G2	高电平选通
4	seg7_data0[7:0]	8bit	输出	数码管字段控制信号输出	D5/B2/B3/A1 B1/A3/A4/B4	高电平有效
5	seg7_data1[7:0]	8bit	输出	数码管字段控制信号输出	H2/D2/E2/F3 F4/D3/E3/D4	高电平有效
6	speed_sel[2:0]	3bit	输入	计数速度选择信号	M4/N4/R1	高电平有效
7	sel_display[2:0]	3bit	输出	计数速度指示	J3/J2/K2	高电平点亮

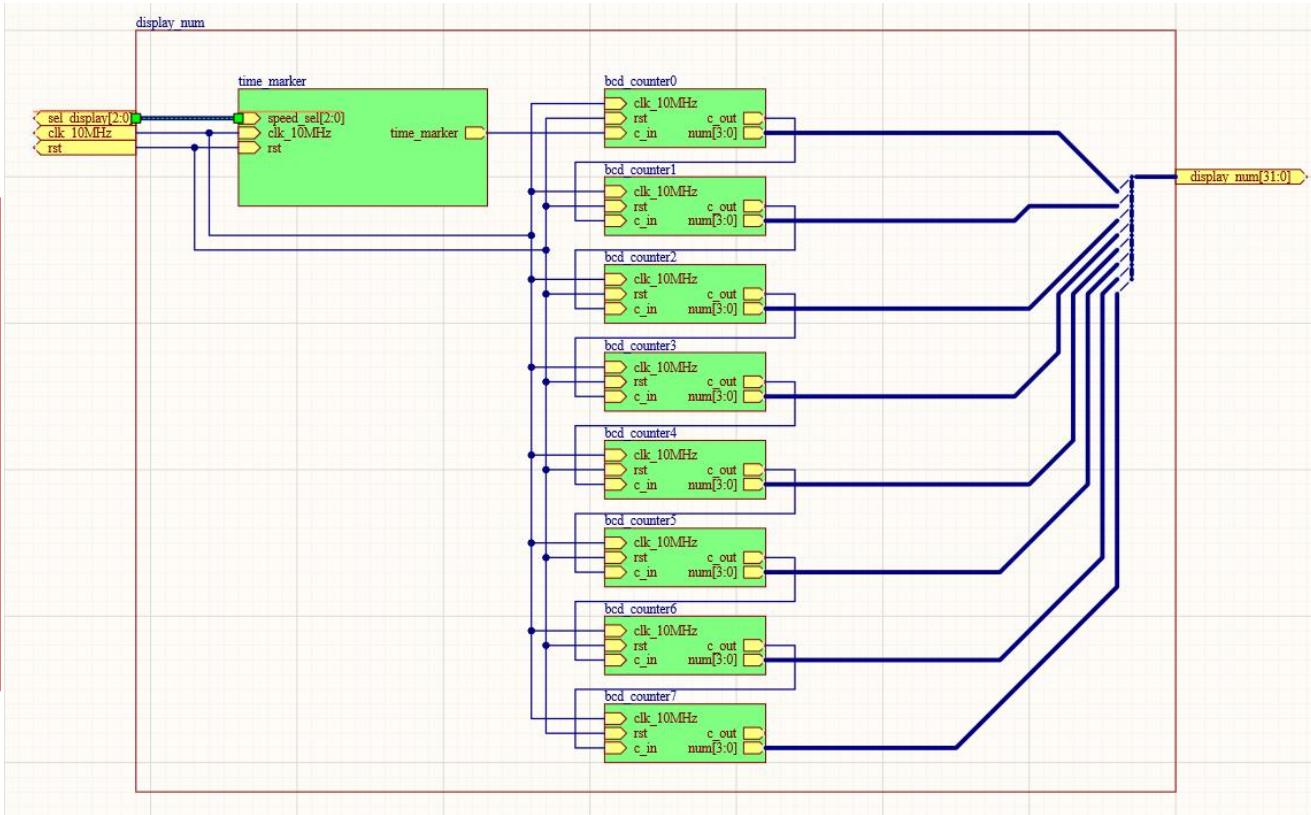
Lab6数码管实验2



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ display_num模块

- 通过time_marker产生计数时基
(时基: 1ms/100ms/1s)
产生一个clk周期的高电平脉冲
- 设计单位bcd_counter
 c_{in} 为进位输入, c_{out} 为进位输出
- 级联后实现8位bcd计数器
最低位bcd_counter以time_marker为进位输入
后级bcd计数器以前级bcd计数器进位输出为进位输入
将各个bcd计数器的输出合并得到计数值



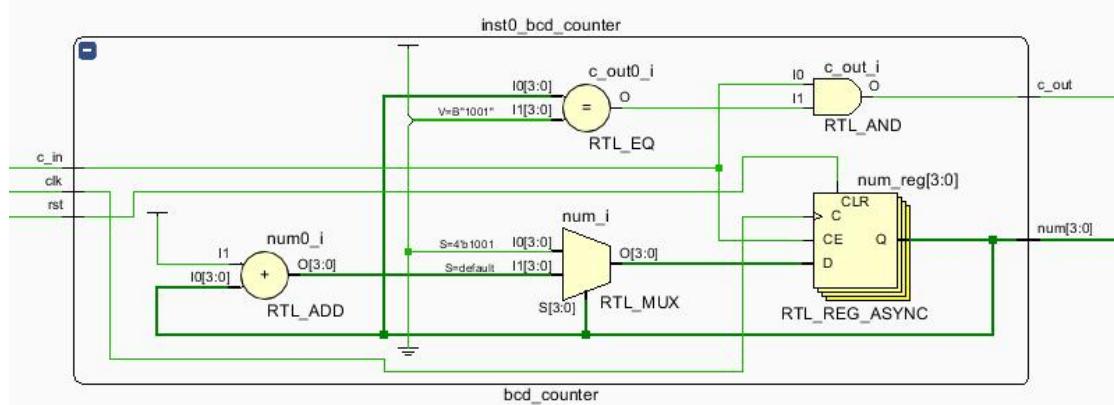
Lab6数码管实验2



```
module bcd_counter(
    input clk,
    input rst,
    input c_in,
    output c_out,
    output reg [3:0] num
);

    always @ (posedge clk or posedge rst) begin
        if(rst) begin
            num   <= 4'd0;
        end
        else if(c_in) begin 时基信号有效，计数+1
            if(num == 4'd9) begin
                num   <= 4'd0;
            end
            else begin
                num   <= num + 1'b1;
            end
        end
        else begin
            num   <= num;
        end
    end
    assign c_out = c_in & (num == 4'd9);
endmodule
```

bcd计数模块(bcd_counter.v)
带进位的1位bcd计数器



Lab6数码管实验2



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

```
module display_num(
    input clk_10MHz,
    input rst,
    input [2:0] speed_sel,
    output [31:0] display_num
);

wire time_marker;
wire cout0,cout1,cout2,cout3,cout4,cout5,cout6;

// generate time marker
time_marker inst_time_marker(
    .clk(clk_10MHz),
    .rst(rst),
    .speed_sel(speed_sel),
    .time_marker(time_marker)
);
// generate LED0 display number
bcd_counter inst0_bcd_counter(
    .clk(clk_10MHz),
    .rst(rst),
    .c_in(time_marker),
    .c_out(cout0),
    .num(display_num[3:0])
);

// generate LED1 display number
bcd_counter inst1_bcd_counter(
    .clk(clk_10MHz),
    .rst(rst),
    .c_in(cout0),
    .c_out(cout1),
    .num(display_num[7:4])
);

// generate LED2 display number
bcd_counter inst2_bcd_counter(
    .clk(clk_10MHz),
    .rst(rst),
    .c_in(cout1),
    .c_out(cout2),
    .num(display_num[11:8])
);
```

```
// generate LED3 display number
bcd_counter inst3_bcd_counter(
    .clk(clk_10MHz),
    .rst(rst),
    .c_in(cout2),
    .c_out(cout3),
    .num(display_num[15:12])
);

// generate LED4 display number
bcd_counter inst4_bcd_counter(
    .clk(clk_10MHz),
    .rst(rst),
    .c_in(cout3),
    .c_out(cout4),
    .num(display_num[19:16])
);

// generate LED5 display number
bcd_counter inst5_bcd_counter(
    .clk(clk_10MHz),
    .rst(rst),
    .c_in(cout4),
    .c_out(cout5),
    .num(display_num[23:20])
);

// generate LED6 display number
bcd_counter inst6_bcd_counter(
    .clk(clk_10MHz),
    .rst(rst),
    .c_in(cout5),
    .c_out(cout6),
    .num(display_num[27:24])
);

// generate LED7 display number
bcd_counter inst7_bcd_counter(
    .clk(clk_10MHz),
    .rst(rst),
    .c_in(cout6),
    .num(display_num[31:28])
);
endmodule
```

计数器模块(display_num.v)
产生显示所需的数据

```
module time_marker(
    input clk,
    input rst,
    input [2:0] speed_sel,
    output reg time_marker
);

reg [31:0] counter = 32'd0;

// reg [31:0] counter_max = 32'd10_000;

// generate counter_max according to speed_sel
// speed_sel == 001, Time Marker = 1ms, counter_max = 1ms/100nm = 10_000
// speed_sel == 010, Time Marker = 100ms, counter_max = 100ms/100nm = 1_000_000
// speed_sel == 100, Time Marker = 1s, counter_max = 1s/100nm = 10_000_000
always @(posedge clk or posedge rst) begin
    if(rst) begin
        counter_max <= 32'd10_000;
    end
    else begin
        case(speed_sel)
            3'b001: counter_max = 32'd10_000;
            3'b010: counter_max = 32'd1_000_000;
            3'b100: counter_max = 32'd10_000_000;
            default: counter_max = 32'd10_000_000;
        endcase
    end
end

//generate time marker
always @(posedge clk or posedge rst ) begin
    if(rst) begin
        counter <= 32'd0;
        time_marker <= 1'b0;
    end
    else if(counter >= (counter_max-1)) begin
        counter <= 32'd0;
        time_marker <= 1'b1;
    end
    else begin
        counter <= counter + 1'b1;
        time_marker <= 1'b0;
    end
end
endmodule
```

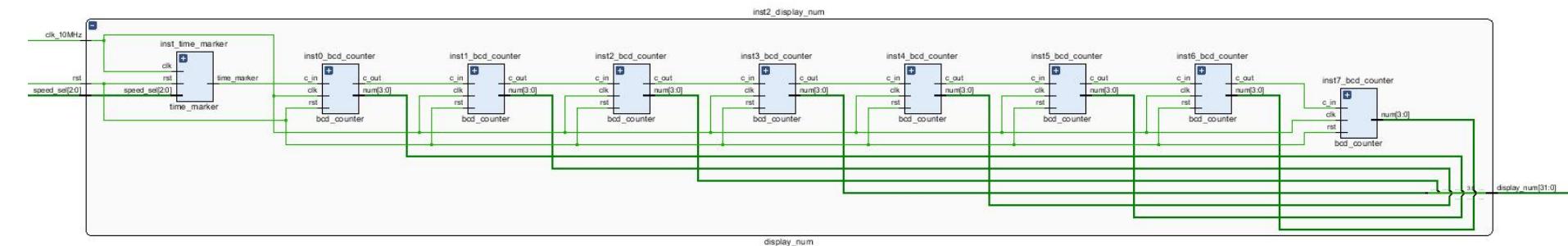
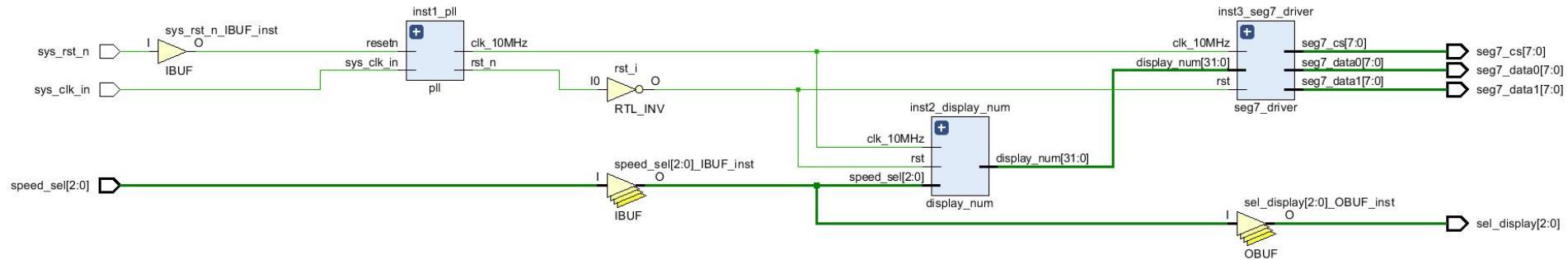
根据速度选择信号
确定计数器的最大值

产生一个clk周期的高电平脉冲time_marker信号

Lab6数码管实验2



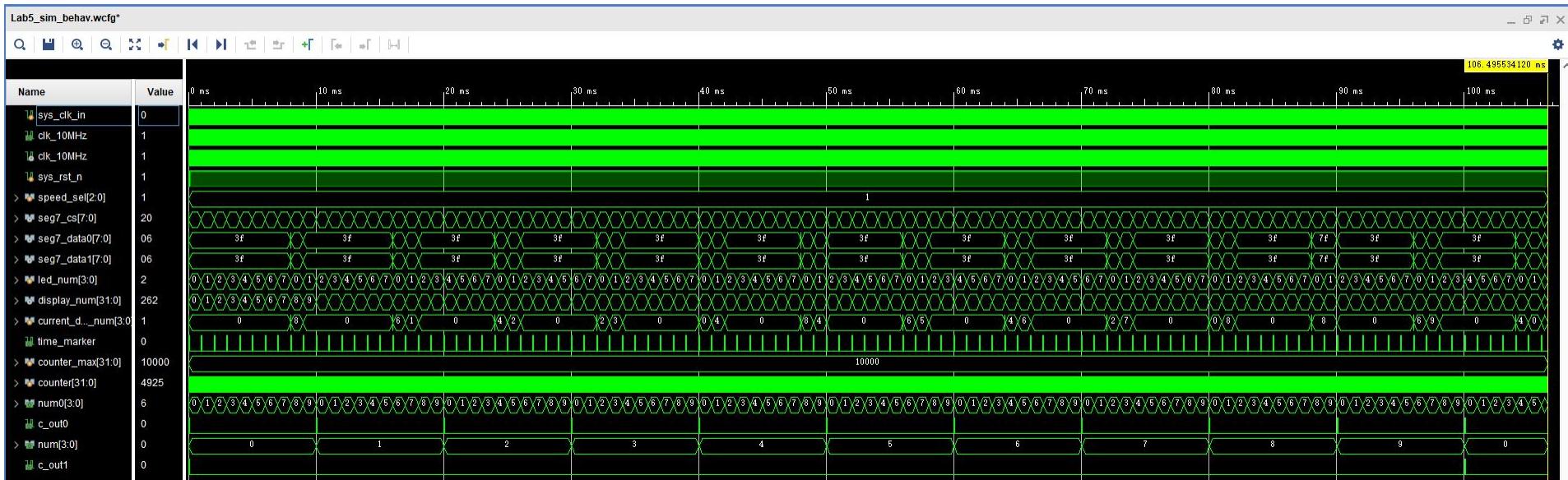
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



Lab6数码管实验2



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



功能仿真波形图



时序分析练习

查看电路的时序裕量

尝试跑到更高的主频

课后作业：HDLbits题目



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

- ▶ Getting Started
- ▶ Verilog Language
- ▼ Circuits
 - ▶ Combinational Logic
 - ▼ Sequential Logic
 - ▶ Latches and Flip-Flops
 - ▼ Counters
 - Four-bit binary counter
 - Decade counter
 - Decade counter again
 - Slow decade counter
 - Counter 1-12
 - Counter 1000
 - 4-digit decimal counter
 - 12-hour clock
 - ▶ Shift Registers
 - ▶ More Circuits
 - ▶ Finite State Machines
 - ▶ Building Larger Circuits
 - ▶ Verification: Reading Simulations
 - ▶ Verification: Writing Testbenches
 - ▶ CS450

下节预告



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Lab7实验内容

Lab7数码时钟实验



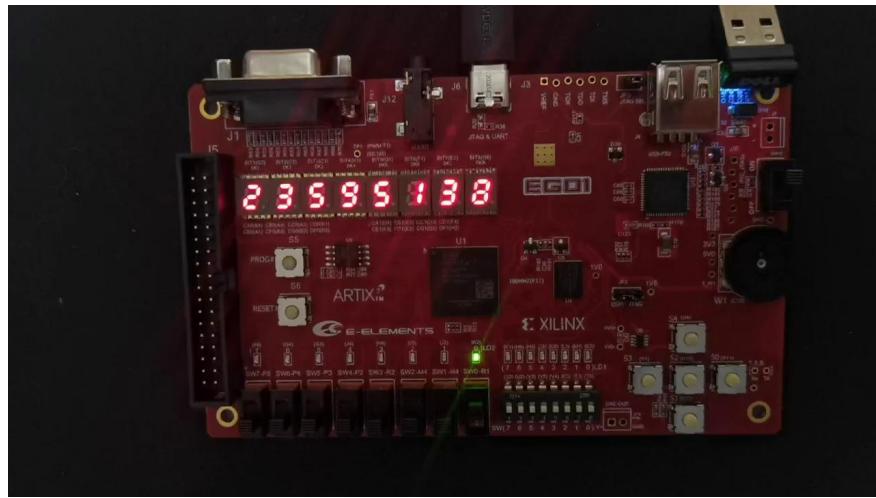
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 实验内容

- 利用已实现的数码管动态显示实验，设计时钟产生模块
- 通过时钟产生模块实现时钟显示
- 实验1：复位时间设置为23(hour)59(min)50(sec)00(ms)，计时至00(hour)00(min)00(sec)00(ms)之后循环，拨码开关控制停止和开始
- 实验2：使用按键实现时间设置功能（秒/分钟/小时设置）

■ 实验目的

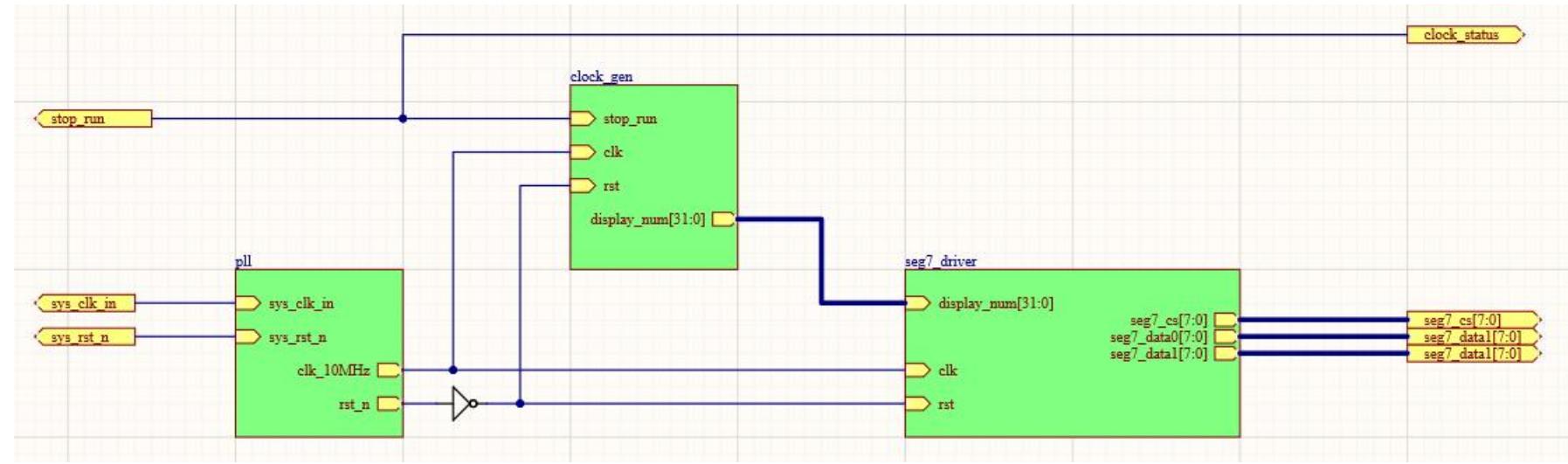
- 熟练使用FPGA层次化结构设计
- 熟悉verilog产生慢速时钟
- 体会按键消抖
- 学会查看简单的时序分析报告



Lab7数码时钟实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



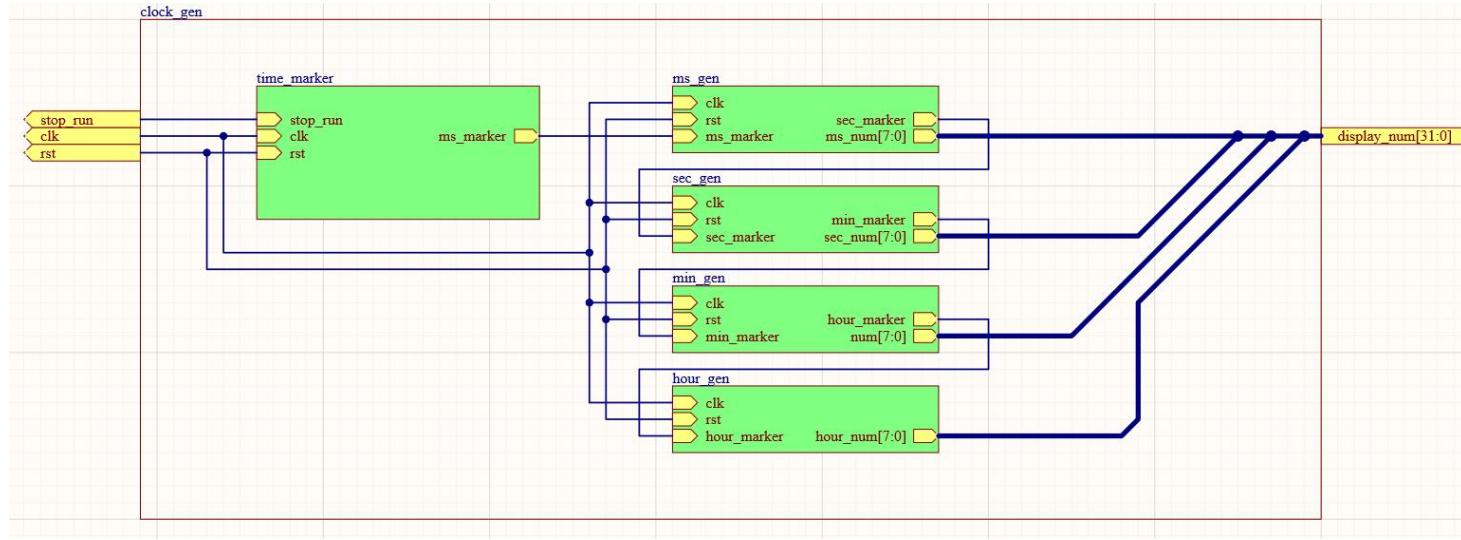
■ 系统框图

- 电路包括三个模块：pll模块实现时钟管理，clock_gen模块产生需要显示的时钟数据，seg7_driver模块完成数码管驱动控制
- pll模块：通过IP核实现，100MHz时钟输入，全局复位引脚输入，生成10MHz信号，频率锁定信号用于后续模块复位
- clock_gen模块：10MHz时钟输入，频率锁定信号作为复位信号，输出需要显示的时钟数据（每位数码管数据用4位表示）
- seg7_driver模块：10MHz时钟输入，频率锁定信号作为复位信号，通过seg7_cs,seg7_data控制数码管点亮

Lab7数码时钟实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



■ clock_gen模块

- time_marker模块产生10ms的计数时基信号，信号特性为1个clk周期的高电平脉冲
- ms_gen模块负责产生毫秒计数值，数值显示区间为00~99，单位10ms
- sec_gen模块负责产生秒计数值，数值显示区间为00~59，单位s
- min_gen模块负责产生分钟计数值，数值显示区间为00~59，单位min
- hour_gen模块负责产生小时计数值，数值显示区间为00~23，单位hour

Lab7数码时钟实验



```
module time_marker(
    input clk,
    input rst,
    input stop_run,
    output reg ms_marker
);

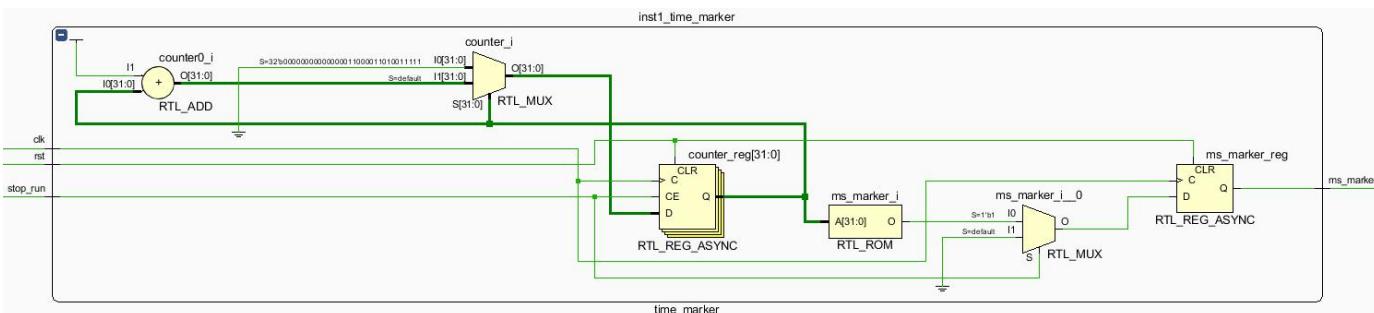
// 定义毫秒时基计数◆?
reg [31:0] counter = 32'd0;

// 定义ms计数器最大值, 10ms/100ns = 100_000;
// 仿真用参数
//parameter COUNTER_10MS = 100_000;

parameter COUNTER_10MS = 100_000;

//产生10ms为时基的脉冲信号
always @(posedge clk or posedge rst) begin
    if(rst) begin
        counter <= 32'd0;
        ms_marker <= 1'b0;
    end
    else if(stop_run) begin
        if(counter == (COUNTER_10MS-1)) begin
            counter <= 32'd0;
            ms_marker <= 1'b1;
        end
        else begin
            counter <= counter + 1'b1;
            ms_marker <= 1'b0;
        end
    end
    else begin
        counter <= counter;
        ms_marker <= 1'b0;
    end
end
endmodule
```

时基产生模块(time_marker.v)
产生10ms脉冲时基信号



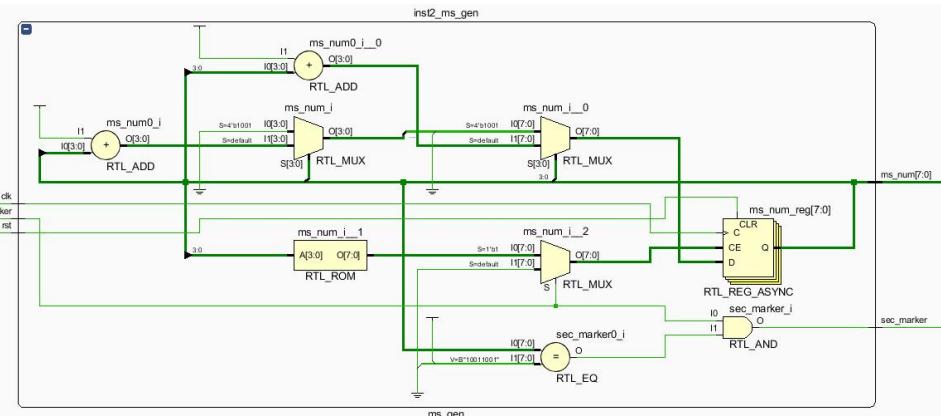
Lab7数码时钟实验



```
module ms_gen(
    input clk,
    input rst,
    input ms_marker,
    output reg [7:0] ms_num
);

    always @ (posedge clk or posedge rst) begin
        if(rst) begin
            ms_num <= 8'd0;
        end
        else if(ms_marker) begin
            if(ms_num[3:0] == 4'd9) begin
                ms_num[3:0] <= 4'd0;
                if(ms_num[7:4] == 4'd9) begin
                    ms_num[7:4] <= 4'd0;
                    ms_num[7:4] <= ms_num[7:4] + 1'b1;
                end
                else begin
                    ms_num[7:4] <= ms_num[7:4] + 1'b1;
                end
            end
            else begin
                ms_num[3:0] <= ms_num[3:0] + 1'b1;
            end
            else begin
                ms_num <= ms_num;
            end
        end
        assign sec_marker = ms_marker & (ms_num == 8'h99); //产生秒脉冲标志
    end
endmodule
```

毫秒计数模块(ms_gen.v)
产生毫秒计数值及秒脉冲



Lab7数码时钟实验



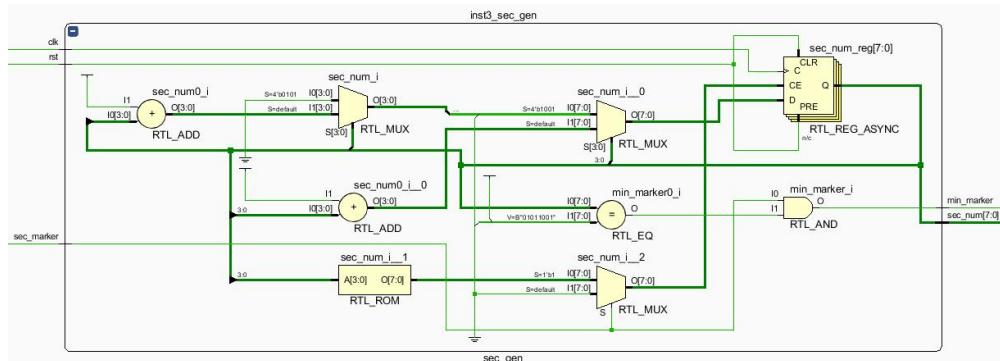
```
module sec_gen(
    input clk,
    input rst,
    input sec_marker,
    output min_marker,
    output reg [7:0] sec_num
);

always @ (posedge clk or posedge rst) begin
    if(rst) begin
        sec_num <= 8'h50;
    end
    else if(sec_marker) begin
        if(sec_num[3:0] == 4'd9) begin
            sec_num[3:0] <= 4'd0;
            if(sec_num[7:4] == 4'd5) begin
                sec_num[7:4] <= 4'd0;
            end
            else begin
                sec_num[7:4] <= sec_num[7:4] + 1'b1;
            end
        end
        else begin
            sec_num[3:0] <= sec_num[3:0] + 1'b1;
        end
    end
    else begin
        sec_num <= sec_num;
    end
end

assign min_marker = sec_marker & (sec_num == 8'h59); //输出分钟脉冲

endmodule
```

秒计数模块(sec_gen.v)
产生秒计数值及分钟脉冲



Lab7数码时钟实验



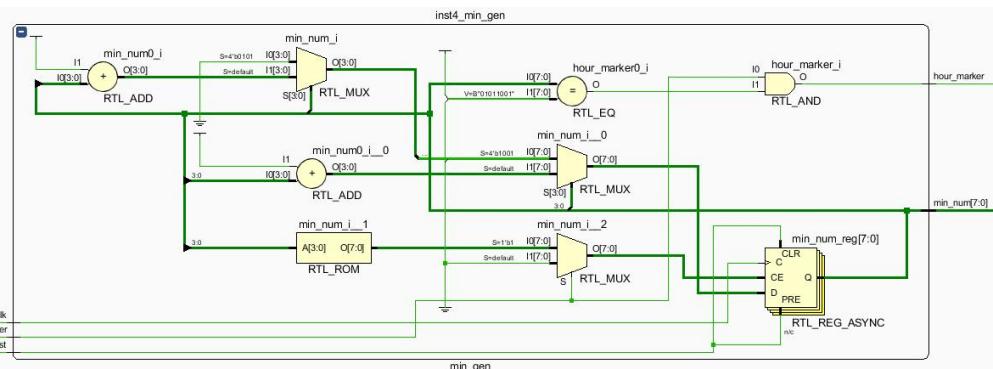
```
module min_gen(
    input clk,
    input rst,
    input min_marker,
    output hour_marker,
    output reg [7:0] min_num
);

always @(posedge clk or posedge rst) begin
    if(rst) begin
        min_num <= 8'h59;
    end
    else if(min_marker) begin
        if(min_num[3:0] == 4'd9) begin
            min_num[3:0] <= 4'd0;
            if(min_num[7:4] == 4'd5) begin
                min_num[7:4] <= 4'd0;
            end
            else begin
                min_num[7:4] <= min_num[7:4] + 1'b1;
            end
        end
        else begin
            min_num[3:0] <= min_num[3:0] + 1'b1;
        end
    end
    else begin
        min_num <= min_num;
    end
end

assign hour_marker = min_marker & (min_num == 8'h59); //产生小时脉冲

endmodule
```

分钟计数模块(min_gen.v)
产生分钟计数值及小时脉冲



Lab7数码时钟实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

```
module hour_gen(
    input clk,
    input rst,
    input hour_marker,
    output reg [7:0] hour_num
);

    always @(posedge clk or posedge rst) begin
        if(rst) begin
            hour_num <= 8'h23;
        end
        else if(hour_marker) begin
            if(hour_num == 8'h23) begin
                hour_num <= 8'd0;
            end
            else if(hour_num[3:0] == 4'd9) begin
                hour_num[3:0] <= 4'd0;
                hour_num[7:4] <= hour_num[7:4] + 1'b1;
            end
            else begin
                hour_num[3:0] <= hour_num[3:0] + 1'b1;
            end
        end
        else begin
            hour_num <= hour_num;
        end
    end
endmodule
```

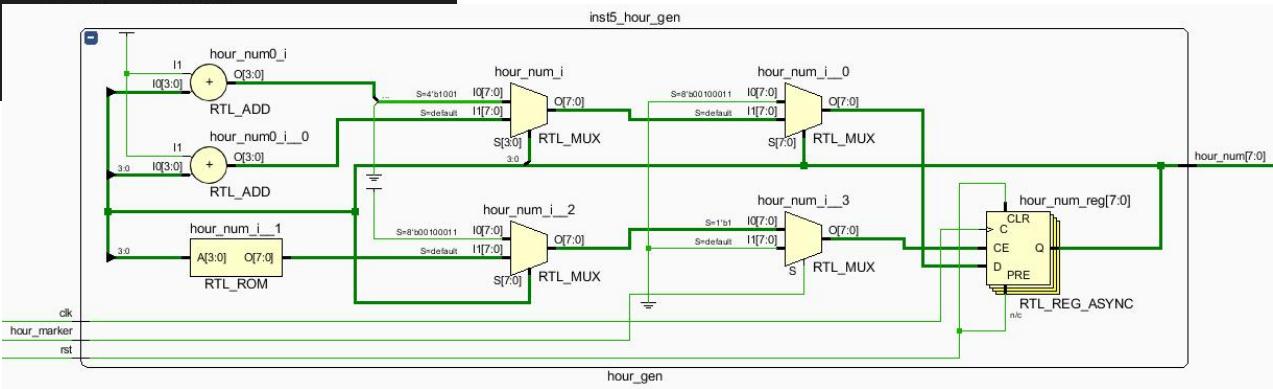
//小时标志到达
//是否为23时，是则清零

//不是23时，个位是否为9，是则个位清零，十位+1

//不是23时，个位不是9，则个位+1

//小时脉冲未到，则保持

小时计数模块(hour_gen.v)
产生小时计数值



Lab7数码时钟实验



```
// 定义数码管刷新周期计数器
reg [31:0] cnt_refresh;
// 定义当前正常刷新的数码管编号
reg [3:0] led_num = 4'd0;

// 刷新计数间隔设置为1ms, 1ms/100ns = 10_000
parameter REFRESH_TIME = 10_000;

// 定义当前显示数据
reg [3:0] current_display_num;

// 产生数码管刷新显示计数器, 每1ms切换一位
always @(posedge clk_10MHz or posedge rst) begin
    if(rst) begin
        cnt_refresh <= 32'd0;
        led_num <= 3'd0;
    end
    else if(cnt_refresh == (REFRESH_TIME-1)) begin
        cnt_refresh <= 32'd0;
        if(led_num == 4'd7) begin
            led_num <= 4'd0;
        end
        else begin      生成数码管刷新计数
            led_num <= led_num + 1'b1;
        end
    end
    else begin
        cnt_refresh <= cnt_refresh + 1'b1;
    end
end
```

```
//根据数码管刷新显示计数器, 选择输出当前的字段与位选信号
always @(posedge clk_10MHz or posedge rst) begin
    if(rst) begin
        current_display_num <= NUM0;
        seg7_cs <= CSN;
    end
    else begin
        case(led_num) 根据数码管刷新计数值
            4'd0: begin
                current_display_num <= display_num[3:0];
                seg7_cs = CS0;
            end
            4'd1: begin
                current_display_num <= display_num[7:4];
                seg7_cs <= CS1;
            end
            4'd2: begin
                current_display_num <= display_num[11:8];
                seg7_cs <= CS2;
            end
            4'd3: begin
                current_display_num <= display_num[15:12];
                seg7_cs <= CS3;
            end
            4'd4: begin
                current_display_num <= display_num[19:16];
                seg7_cs <= CS4;
            end
            4'd5: begin
                current_display_num <= display_num[23:20];
                seg7_cs <= CS5;
            end
            4'd6: begin
                current_display_num <= display_num[27:24];
                seg7_cs <= CS6;
            end
            4'd7: begin
                current_display_num <= display_num[31:28];
                seg7_cs <= CS7;
            end
            default: begin
                current_display_num <= NUM0;
                seg7_cs <= CSN;
            end
        endcase
    end
end
```

数码管驱动模块(seg7_driver.v)
通过字库索引输出数据

```
//根据数码管显示数据进行字库索引
always @(posedge clk_10MHz or posedge rst) begin
    if(rst) begin
        seg7_data0 <= NUM0;
    end
    else begin
        case(current_display_num)
            4'h0: seg7_data0 <= NUM0;
            4'h1: seg7_data0 <= NUM1;
            4'h2: seg7_data0 <= NUM2;
            4'h3: seg7_data0 <= NUM3;
            4'h4: seg7_data0 <= NUM4;
            4'h5: seg7_data0 <= NUM5;
            4'h6: seg7_data0 <= NUM6;
            4'h7: seg7_data0 <= NUM7;
            4'h8: seg7_data0 <= NUM8;
            4'h9: seg7_data0 <= NUM9;
            4'ha: seg7_data0 <= NUMA;
            4'hb: seg7_data0 <= NUMB;
            4'hc: seg7_data0 <= NUMC;
            4'hd: seg7_data0 <= NUMD;
            4'he: seg7_data0 <= NUME;
            4'hf: seg7_data0 <= NUMF;
            default: seg7_data0 <= NUM0;
        endcase
    end
    seg7_data1 <= seg7_data0;
end
```



THANKS



國科大杭州高等研究院
Hangzhou Institute for Advanced Study, UCAS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第九讲 (9.1 课程设计选题)

2025年4月29日

课程介绍

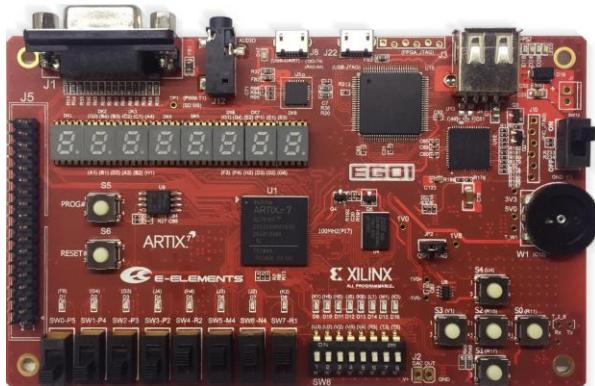


國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 课程教学计划

- 第1讲：课程介绍+数字电路基础+工具使用介绍+**Lab0.0：FPGA入门实验（1）**
- 第2讲：FPGA概述+Vivado工具使用+**Lab0.1：FPGA入门实验（2）**
- 第3讲：FPGA硬件电路+HDL语言概述+**Lab1：LED控制实验**
- 第4讲：Verilog语法（1）+**Lab2：分频计数器**
- 第5讲：Verilog语法（2）+**Lab3：流水灯实验（1）**
- 第6讲：Verilog语法（3）+**Lab4：数码管实验（1）**
- 第7讲：状态机+**Lab5：红绿灯实验**
- 第8讲：FPGA时序分析基础+**Lab6：数码管实验（2）**
- 第9讲：**课程设计选题**+**Lab7：数码时钟实验**
- 第10讲：XADC概述及使用+**Lab8：XADC实验**
- 第11讲：FPGA串口概述+**Lab9：串口数据接收实验**
- 第12讲：FPGA项目开发及管理+**Lab10：串口数据发送实验**
- 第13讲：**课程设计研讨1：项目需求分析讨论**
- 第14讲：**课程设计研讨2：项目概要设计讨论**
- 第15讲：**课程设计研讨3：项目详细设计讨论**
- 第16讲：**课程设计研讨4：板级效果展示+项目总结**
- **期末考试：10道题目开卷考试（2小时）**

怎么学？





1. 数码时钟
2. DDS信号发生器
3. 逻辑分析仪
4. 数字频率计
5. 信号滤波器
6. 其他

1. 数码时钟



■ 功能描述

- 24小时时钟显示，显示到秒级
- 闹钟功能
- 秒表功能，显示到毫秒
- 三键式菜单，实现数码时钟的各种功能设置
- 扩展功能：带有通讯控制功能（串口设置时间，串口回读时间等）

■ 所需硬件资源

- EGo1板载数码管，按键



2.DDS信号发生器



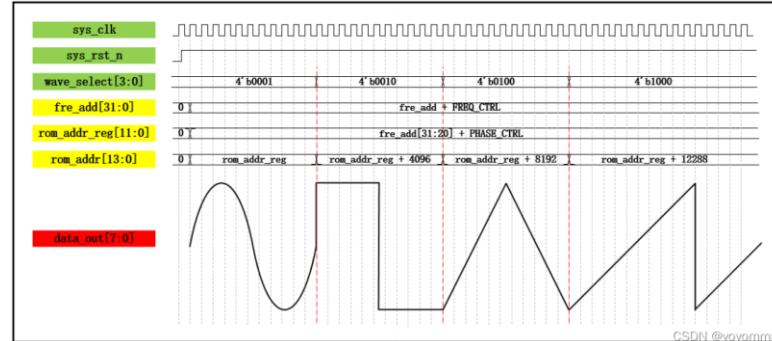
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 功能描述

- 可输出方波、三角波、正弦波等三种波形
- 通过串口设置各种波形的参数，包括波形模式、频率及幅度等
- 自定义最高频率、频率稳定度等技术指标

■ 所需硬件资源

- 高速DAC板卡（芯片）：AD9767
- 专用DDS板卡（芯片）：AD9850、AD9910等
- 示波器、频谱分析仪等



ANALOG
DEVICES

10-/12-/14-Bit, 125 MSPS
Dual TxDAC+ Digital-to-Analog Converters

AD9763/AD9765/AD9767

Data Sheet

FEATURES

10-/12-/14-bit dual transmit digital-to-analog converters (DACs)
125 MSPS update rate
Excellent SFDR to Nyquist @ 5 MHz output: 75 dBc
Excellent gain and offset matching: 0.1%
Fully independent or single-resistor gain control
Dual-port or interleaved data
On-chip 1.2 V reference
5 V or 3.3 V operation
Power dissipation: 380 mW @ 5 V
Power-down mode: 50 mW @ 5 V
48-lead LQFP

APPLICATIONS

Communications
Base stations
Digital synthesis
Quadrature modulation
3D ultrasound

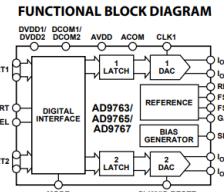


Figure 1.

The DACs utilize a segmented current source architecture combined with a proprietary switching technique to reduce glitch energy and maximize dynamic accuracy. Each DAC provides differential current output, thus supporting single-ended or differential applications. Both DACs of the AD9763, AD9765, or

3.逻辑分析仪

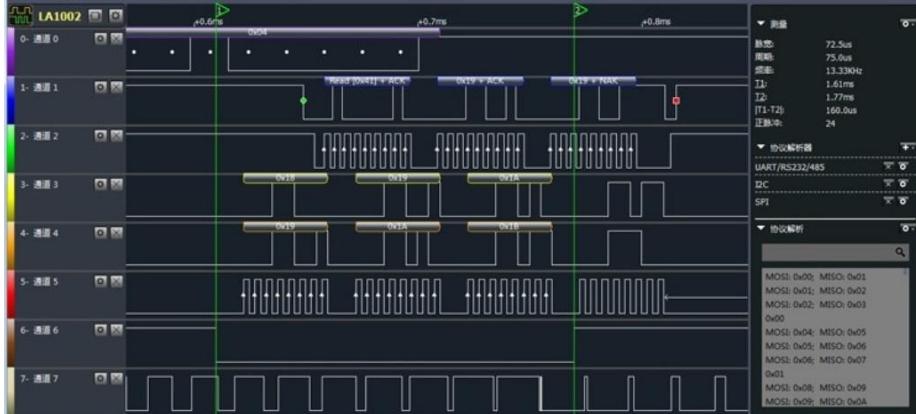


■ 功能描述

- 高速采集数字信号
- 将数据暂存于内部存储区
- 通过串口将数据发送到上位机显示 (或VGA显示)
- 扩展功能：使用高速ADC，实现简易数字示波器功能

■ 所需硬件资源

- 高速ADC板卡 (芯片) : AD9226
- SDRAM模块
- 信号发生器



逻辑分析仪主要规格：
采集时长和采样率



4.数字频率计



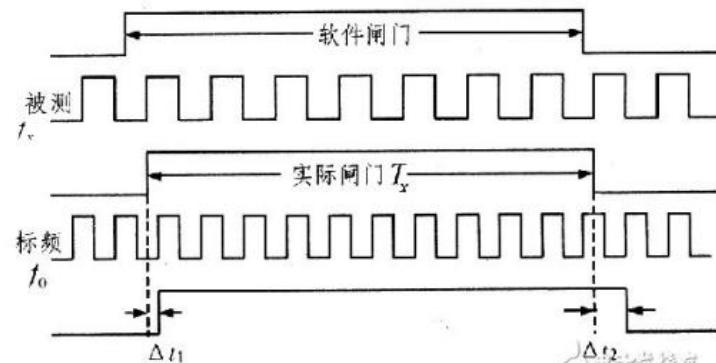
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 功能描述

- 高精度测量待测信号（数字信号）的频率
- 测频结果显示在数码管上
- 支持通过串口将测频结果回传至计算机

■ 所需硬件资源

- EGo1板载数码管
- 信号发生器



5.信号滤波器



■ 功能描述

- 通过ADC采集带有噪声的模拟信号
- 设计滤波器模块对该模拟信号进行数字滤波处理
- 通过高精度DAC输出滤波后的模拟信号
- 滤波器参数可由串口进行设置

■ 所需硬件资源

- EGo1 XADC
- 高速高精度ADC板卡（芯片）：AD9226
- 高精度DAC板卡（芯片）：AD9767
- 信号发生器

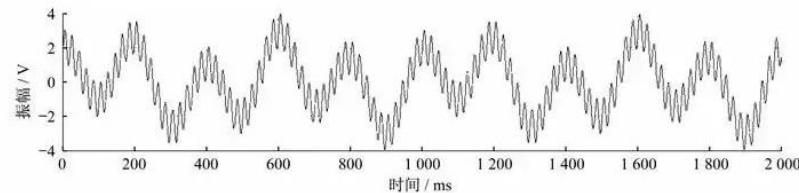


图 3 输入信号波形
Fig. 3 Input signal waveform

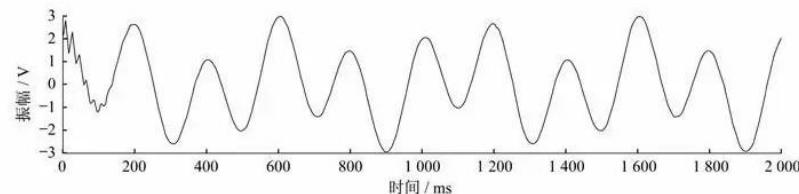


图 4 输出信号波形
Fig. 4 Output signal waveform

6.其他选题



- 摄像头数据采集
- 出租车计价器
- 音乐播放器
- 游戏类：贪吃蛇、俄罗斯方块
- 控制类：数控控温系统
- 通信类：对讲机
- ...



FPGA竞赛选题



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

集创赛

一起从芯出发

第九届全国大学生集成电路创新创业大赛

大赛简介

国内集成电路领域最大规模高校赛事，中国高等教育学会全国高校竞赛榜单唯一上榜集成电路专业赛事

企业怀责，覆盖全产业链

三大奖项，面向多层次人才

产学合作，全方位生态服务

百万奖金

<http://univ.ciciec.com/>

全国大学生嵌入式芯片与系统设计竞赛—FPGA创新设计赛道

首页 竞赛说明 通知公告 常见问题 培训视频 参赛院校 往届回顾 企业专区 获奖信息 下载中心 联系我们 登录 注册

2024年FPGA创新设计 赛道通知发布

查看更多

主办单位 中国电子教育学会

承办单位 东南大学
南京市江北新区管委会

协办单位 AMD
上海海信信息科技股份有限公司
易见思（深圳）科技有限公司
广东嘉云半导体科技股份有限公司
深圳市紫光同创有限公司
信息技术新工科产学研联盟司定制计算工作委员会

支持单位 南京集成电路产业服务中心（ICsC）
运营单位 南京集成电路培训基地

<http://www.fpgachina.cn/index>

FPGA竞赛选题-集创赛



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

一起从芯出发

第九届全国大学生集成电路创新创业大赛

大赛简介

国内集成电路领域最大规模高校赛事，中国高等教育学会全国高校竞赛榜单唯一入榜集成电路专业赛事

企业杯赛，覆盖全产业链

三大赛项，面向多层次人才

产学合作，全方位生态服务

百万奖金

创业成...

主赛项、职业技能赛项、创新创业成果赛项，面向本硕博、中高职不同层次人才

人才招聘、成果转化、协同育人、赛事服务平台助力产学研全方位合作

全国大学生集成电路创新创业大赛
CHINA COLLEGE IC COMPETITION

关于大赛 主赛项参赛 职业技能赛项 新闻通知

第九届集创赛杯赛题目

本届大赛共分为4大技术方向，11个赛道，25个杯赛，请选择你感兴趣的赛题报名参赛吧！

芯片设计方向

芯片应用...

半导体产业...

创业成...

FPGA应用开发

处理器应用

紫光同创杯
中科亿海微杯
海云捷迅杯

算能杯
飞腾杯
奕斯伟杯

<http://univ.ciciec.com/>

FPGA竞赛选题-集创赛-紫光同创



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

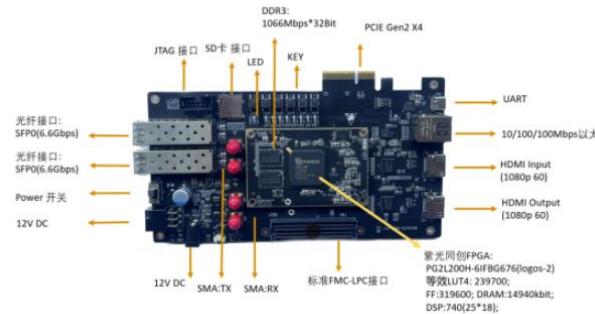
■ 基于紫光同创FPGA的机器人

➤ 功能描述

- 机器人运动控制（控制机器人简单移动）
- 传感器数据处理（实现环境感知）
- 人机交互接口（通过按键、上位机等实现对机器人的控制）

➤ 所需硬件资源

- 盘古676系列开发板
- 盘古50开发板 (MES50HP)



http://univ.ciciec.com/nd.jsp?id=888#_jcp=1



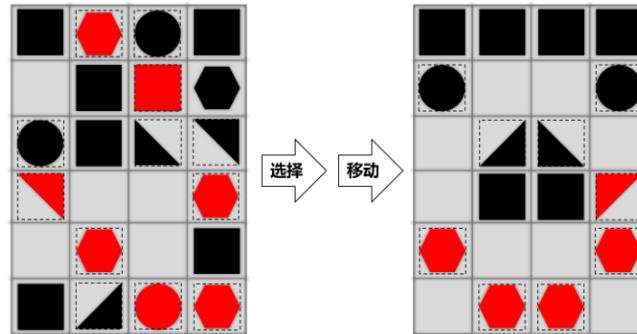
■ 基于FPGA的机械臂精确控制系统设计

➤ 功能描述

- 机械臂运动控制（完成多自由度机械臂控制）
- 传感器数据处理（实现对不同颜色、形状物块的识别）
- 人机交互接口（通过按键、上位机等实现对机器人的控制）

➤ 所需硬件资源

- AWC_C4 FPGA





■ 基于中科亿海微FPGA的图像处理系统

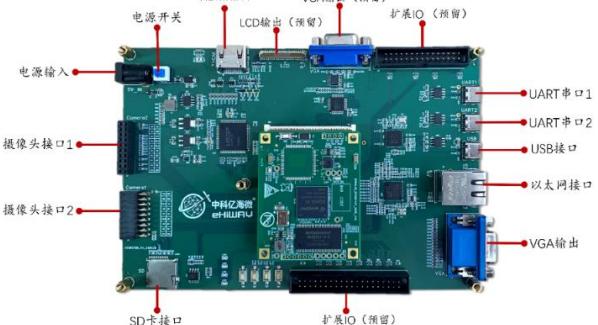
➤ 功能描述

- 图像采集（使用摄像头完成对图像的事实采集）
- 图像处理（实现对目标物体特征的提取、识别、分类）
- 图像显示（通过VGA或HDMI将视频图像与识别结果实时显示）

➤ 所需硬件资源

• 摄像头

• Link-Sea-H6A图像处理套件



FPGA竞赛-集创赛-获奖证书



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA竞赛-大学生嵌入式FPGA赛道



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

全国大学生嵌入式芯片与系统设计竞赛
FPGA创新设计赛道

National Undergraduate Embedded Chip and System Design Competition - FPGA Innovation Design Track

竞赛说明 COMPETITION DESCRIPTION

竞赛背景

全国大学生嵌入式芯片与系统设计竞赛（以下简称“大赛”）旨在提高全国高校学生在嵌入式芯片及系统设计领域、可编程逻辑器件应用领域自主创新设计与工程实践能力，培养具有创新思维、具备解决复杂工程问题能力且拥有团队合作精神的优秀人才，推进高校与企业人才培养合作共建。大赛强调在做芯片与系统设计的同时关注可编程逻辑器件应用领域创新设计原理及规律，鼓励参赛师生掌握FPGA相关的理论知识和实践技能，丰富和活跃高校创新创业学术氛围。

本届大赛设芯片应用赛道、芯片设计赛道和FPGA创新设计赛道，彼此相互独立。FPGA创新设计赛道秉承嵌入式大赛宗旨，旨在提高学生在数字系统设计领域尤其是可编程逻辑器件应用领域的创新实践能力。

全国大学生FPGA创新设计赛道

首页 竞赛说明 通知公告 常见问题 培训视频 参赛院校 往届回顾 企业专区 获奖信息 下载中心 联系我们

登录 注册

首页 > 下载中心

FPGA仲裁申请表 NEW 2024-11-20 下载

大赛通知 | 关于邀请参加2024年全国大学生嵌入式芯片与系统设计竞赛——FPGA创新设计赛道 ... NEW 2024-09-04 下载

2024年FPGA创新设计赛道选题指南-AMD NEW 2024-09-05 下载

2024年FPGA创新设计赛道选题指南-安路科技 NEW 2024-09-04 下载

2024年FPGA创新设计赛道选题指南-易灵思 NEW 2024-09-04 下载

2024年FPGA创新设计赛道选题指南-高云半导体 NEW 2024-09-05 下载

2024年FPGA创新设计赛道选题指南-紫光同创 NEW 2024-09-05 下载

<http://www.fpgachina.cn/index>



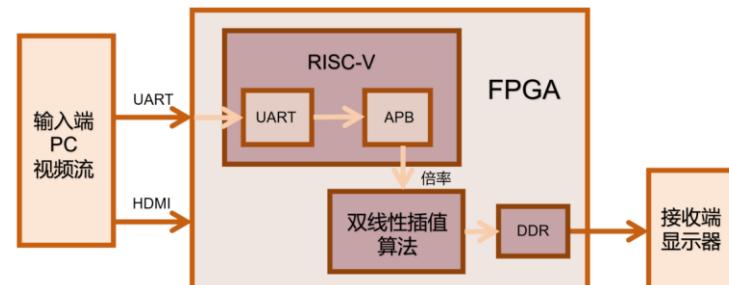
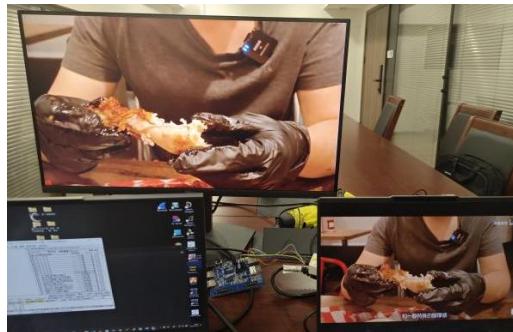
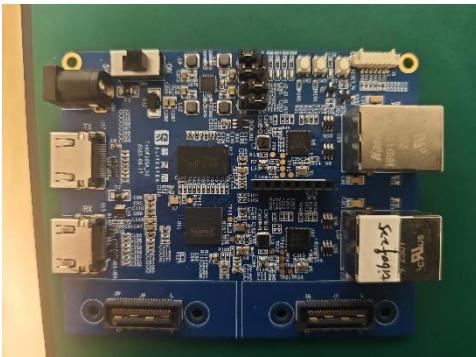
■ 基于 Ti60F225 无极缩放算法实现

➤ 功能描述

- 基于HDMI实现视频流的实时传输 (640*480)
- 利用双线性插值类算法实现视频流的缩放
- 通过按键控制视频缩放倍率的加减

➤ 所需硬件资源

- 易灵思Ti60F225
- 摄像头

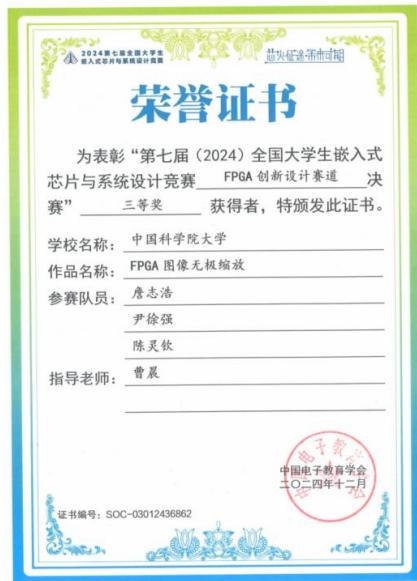
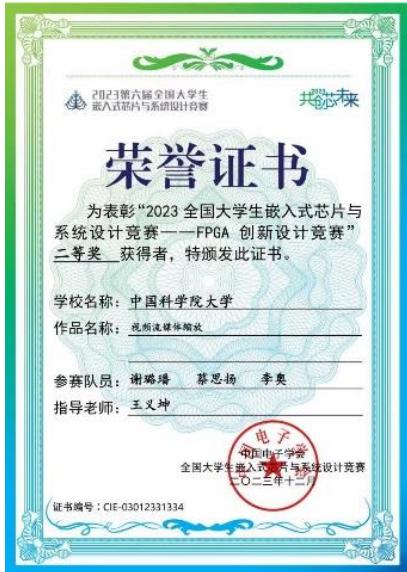


<http://www.fpgachina.cn/download?value=11>

FPGA竞赛-大学生嵌入式FPGA赛道-获奖证书



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA设计方法



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Bottom-up

Top-down

系统分解

行为设计

单元设计

结构设计

功能块划分

逻辑设计

子系统设计

电路设计

系统总成

版图设计

自底向上设计

自顶向下设计

FPGA设计采用**自顶向下设计**

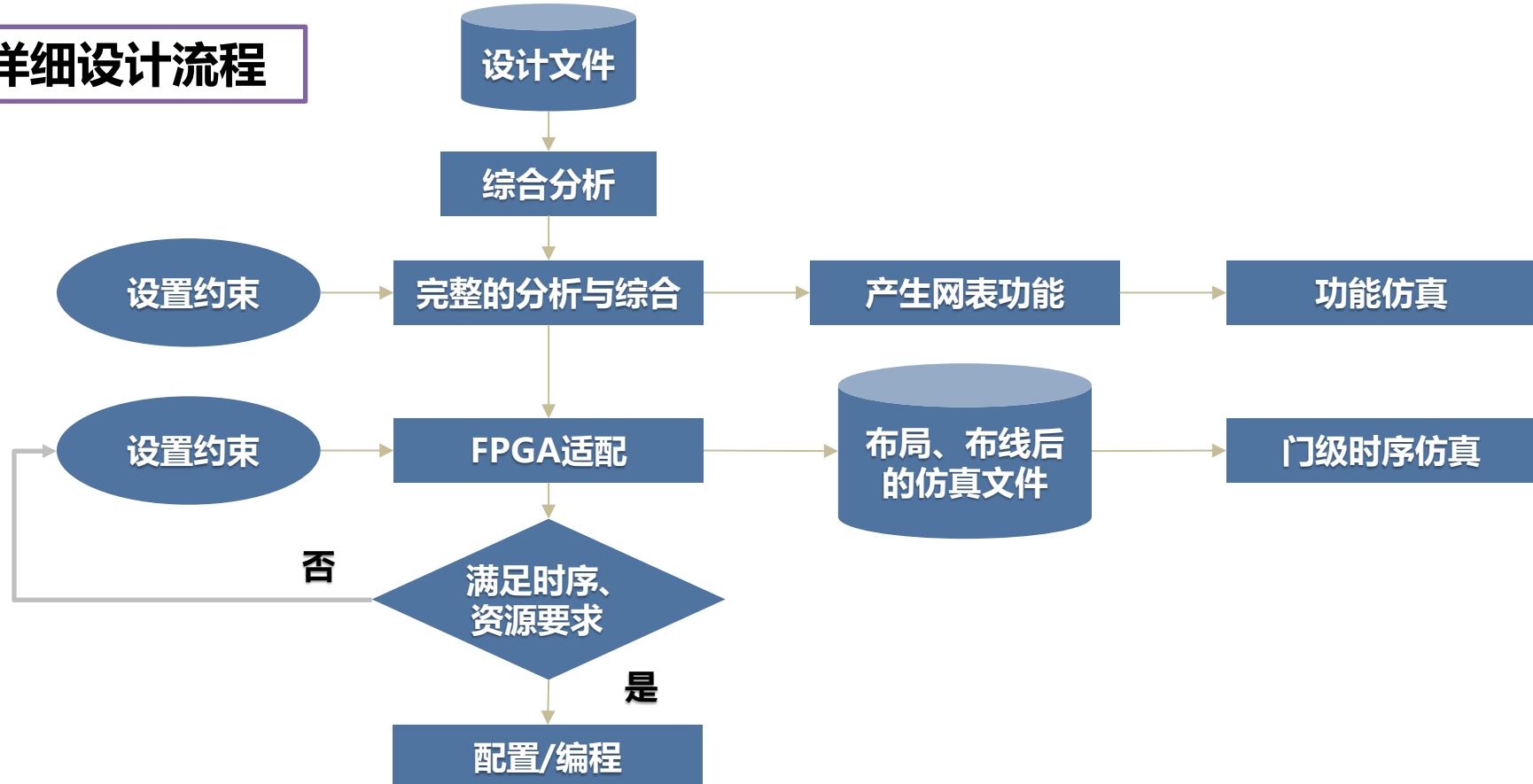
- 将硬件系统划分成模块
 - 设计出各模块的行为功能或结构
 - 进行仿真以检验设计是否正确
 - 将模块分给下一级设计者
-
- 更加的符合人们的逻辑思维习惯
 - 便于设计者对复杂的系统进行划分与优化

FPGA设计方法



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

详细设计流程



FPGA项目开发流程



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

需求分析

概要设计

详细设计

项目总结

功能需求

性能需求

外部接口需求

资源需求

设计约束

其他设计需求

系统组成

模块划分

子模块方案

关键技术分析

方案所需资源分析

系统组成及
输入输出分析

系统模块划分
及接口分析

子模块详细设计

关键代码实现
及分析

系统及模块
测试方案

功能完成情况

技术指标完成情况

关键代码实现
与分析

系统仿真情况

代码调试情况

实现效果



FPGA项目开发设计报告提纲



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

FPGA 电路软硬件设计

课程设计需求分析报告

一、引言

- 1.1 编写目的
- 1.2 课题背景
- 1.3 术语定义
- 1.4 参考资料

二、概述

- 2.1 功能描述
- 2.2 性能描述
- 2.3 开发环境
- 2.4 设计约束

三、详细需求分析

- 3.1 功能需求
- 3.2 性能需求
- 3.3 外部接口需求
- 3.4 所需资源需求
- 3.5 设计约束
- 3.6 其他设计需求

四、总结

FPGA 电路软硬件设计

课程设计概要设计报告

一、引言

- 1.1 编写目的
- 1.2 术语定义
- 1.3 参考资料

二、概述

- 2.1 设计要求
- 2.2 总体设计思路

三、总体方案

- 3.1 系统组成
- 3.2 模块划分
- 3.3 子模块 1 方案
- 3.4 子模块 2 方案
- 3.5 关键技术分析
- 3.6 方案所需资源分析

四、总结



FPGA 电路软硬件设计

课程设计详细设计报告

一、引言

- 1.1 编写目的
- 1.2 术语定义
- 1.3 参考资料

二、概述

- 2.1 设计要求
- 2.2 总体方案概述

三、详细设计

- 3.1 系统组成及输入输出分析
- 3.2 系统模块划分及接口分析
- 3.3 模块 1 详细设计
- 3.4 模块 2 详细设计
- 3.5 关键代码实现及分析
- 3.6 系统及模块测试方案

四、总结

FPGA 电路软硬件设计

课程设计总结报告

一、引言

- 1.1 编写目的
- 1.2 术语定义
- 1.3 参考资料

二、概述

- 2.1 课程设计概述

三、课程设计完成情况

- 3.1 课程设计的功能完成情况
- 3.2 课程设计的技术指标完成情况
- 3.3 关键代码实现及分析
- 3.4 系统仿真情况
- 3.5 代码调试情况
- 3.6 实现效果

四、总结

- 4.1 存在的主要问题
- 4.2 后续的改进方向

选题及研讨要求



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

✓ 选题组队要求：

- **自由选题；**
- **选课49人，组成12队，每个团队至少4人，自由组队；**
- **两周之内完成，5月13日18: 00前提交课程设计选题表；**

✓ 研讨课要求：

- **4个开发阶段每组轮流派代表介绍工作，每次课堂分享成员不同，时间不超过10分钟，格式不限；**
- **研讨课当周的周五18: 00前提交修改完善后的阶段报告（pdf格式），要求内容完整、格式规范。**



THANKS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第九讲 (9.2 数码时钟实验)

2025年4月29日

FPGA课程设计选题-数码时钟



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 功能描述

- 24小时时钟显示，显示到秒级
- 闹钟功能
- 秒表功能，显示到毫秒
- 三键式菜单，实现数码时钟的各种功能设置
- 扩展功能：带有通讯控制功能（串口设置时间，串口回读时间等）

■ 所需硬件资源

- EGo1板载数码管，按键



Lab7数码时钟实验



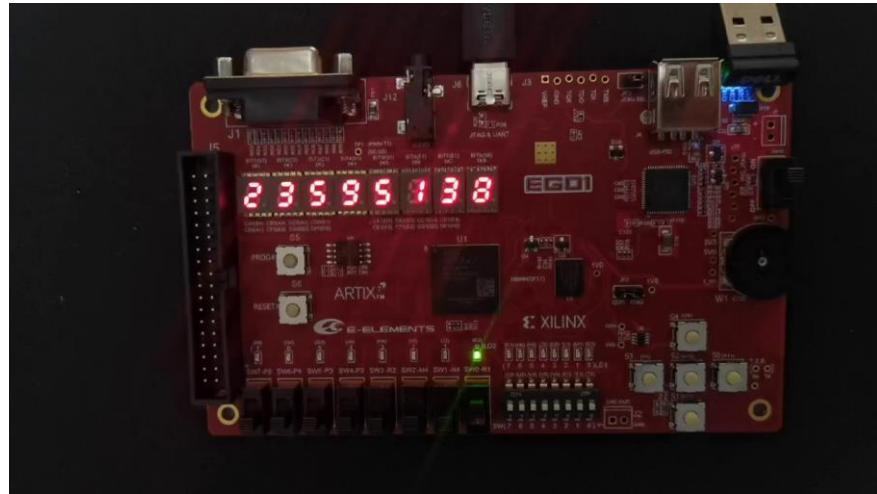
國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 实验内容

- 利用已实现的数码管动态显示框架，设计时钟产生模块
- 通过时钟产生模块实现时钟信号生成
- 实验1：复位时间设置为23(hour)59(min)50(sec)00(ms)，计时至00(hour)00(min)00(sec)00(ms)之后循环，拨码开关控制停止和开始
- 实验2：使用按键实现时间设置功能（秒/分钟/小时设置）

■ 实验目的

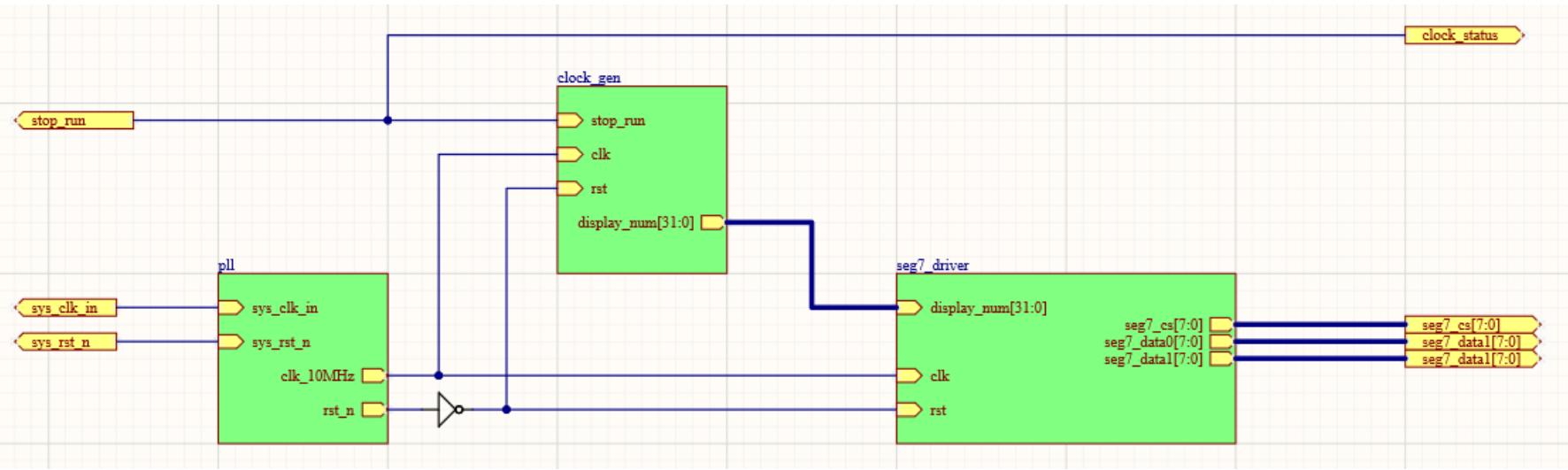
- 熟练使用FPGA层次化结构设计
- 熟悉verilog产生慢速时钟
- 体会按键消抖
- 学会查看简单的时序分析报告



Lab7数码时钟实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



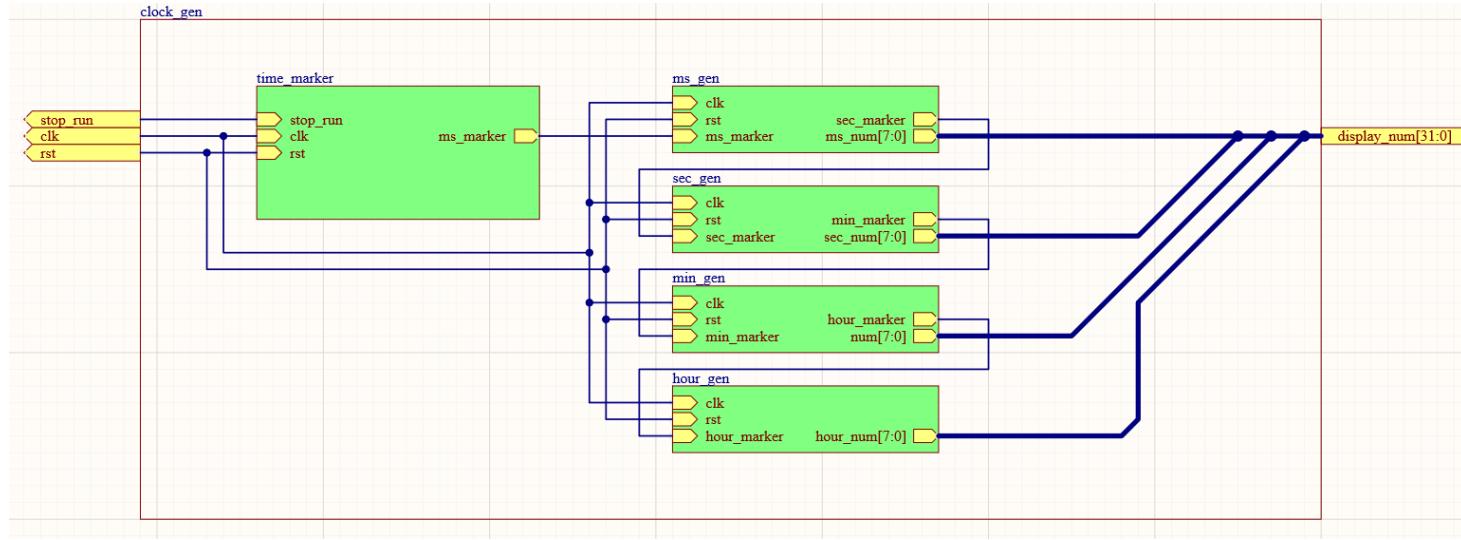
■ 系统框图

- 电路包括三个模块：pll模块实现时钟管理，clock_gen模块产生需要显示的时钟数据，seg7_driver模块完成数码管驱动控制
- pll模块：通过IP核实现，100MHz时钟输入，全局复位引脚输入，生成10MHz信号，频率锁定信号用于后续模块复位
- clock_gen模块：10MHz时钟输入，频率锁定信号作为复位信号，输出需要显示的时钟数据（每位数码管数据用4位表示）
- seg7_driver模块：10MHz时钟输入，频率锁定信号作为复位信号，通过seg7_cs,seg7_data控制数码管点亮

Lab7数码时钟实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



■ clock_gen模块

- time_marker模块产生10ms的计数时基信号，信号特性为1个clk周期的高电平脉冲
- ms_gen模块负责产生毫秒计数值，数值显示区间为00~99，单位10ms
- sec_gen模块负责产生秒计数值，数值显示区间为00~59，单位s
- min_gen模块负责产生分钟计数值，数值显示区间为00~59，单位min
- hour_gen模块负责产生小时计数值，数值显示区间为00~23，单位hour

Lab7数码时钟实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

输入输出端口表

序号	信号名称	信号类型	输入/输出	信号描述	FPGA引脚	说明
1	sys_clk_in	1bit	输入	系统时钟输入	P17	100MHz输入
2	sys_RST_n	1bit	输入	系统复位输入	P15	低电平复位
3	stop_run	1bit	输入	时钟使能输入	R1	高电平使能有效
4	clock_status	1bit	输出	时钟状态显示	K2	高电平点亮，代表时钟运行
5	seg7_cs[7:0]	8bit	输出	数码管位选控制信号输出	G2/C2/C1/H1 G1/F1/E1/G6	高电平选通
6	seg7_data0[7:0]	8bit	输出	数码管字段控制信号输出	B4/A4/A3/B1 A1/B3/B2/D5	高电平有效
7	seg7_data1[7:0]	8bit	输出	数码管字段控制信号输出	D4/E3/D3/F4 F3/E2/D2/H2	高电平有效

Lab7数码时钟实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

```
module Lab6_top(
    input sys_clk_in,
    input sys_rst_n,
    input stop_run,
    output clock_status,
    output [7:0] seg7_cs,
    output [7:0] seg7_data0,
    output [7:0] seg7_data1
);

/////////////////////////////////////////////////////////////////
wire clk_10MHz;
wire rst_n;
wire rst;
wire [31:0] display_num;

/////////////////////////////////////////////////////////////////
assign rst = ~rst_n;
assign clock_status = stop_run;

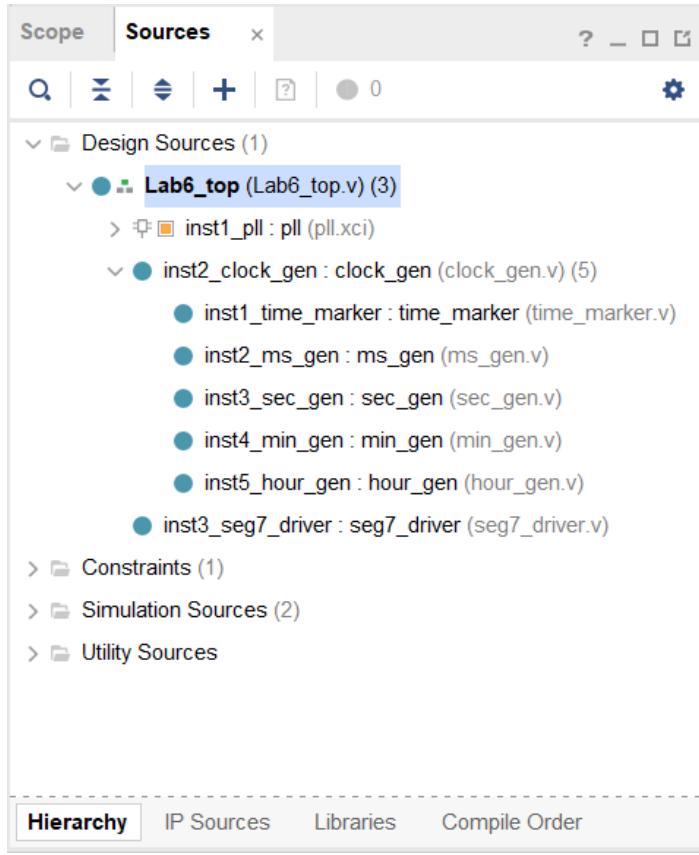
/////////////////////////////////////////////////////////////////
pll inst1_pll
(
    .clk_10MHz(clk_10MHz),           // output clk_10MHz
    .resetn(sys_rst_n),              // input resetn
    .rst_n(rst_n),                  // output rst_n
    .sys_clk_in(sys_clk_in)          // input sys_clk_in
);

/////////////////////////////////////////////////////////////////
clock_gen inst2_clock_gen
(
    .clk(clk_10MHz),
    .rst(rst),
    .stop_run(stop_run),
    .display_num(display_num)
);

/////////////////////////////////////////////////////////////////
seg7_driver inst3_seg7_driver
(
    .clk(clk_10MHz),
    .rst(rst),
    .display_num(display_num),
    .seg7_cs(seg7_cs),
    .seg7_data0(seg7_data0),
    .seg7_data1(seg7_data1)
);

/////////////////////////////////////////////////////////////////
endmodule
```

顶层文件(Lab7_top.v)
完成模块互联

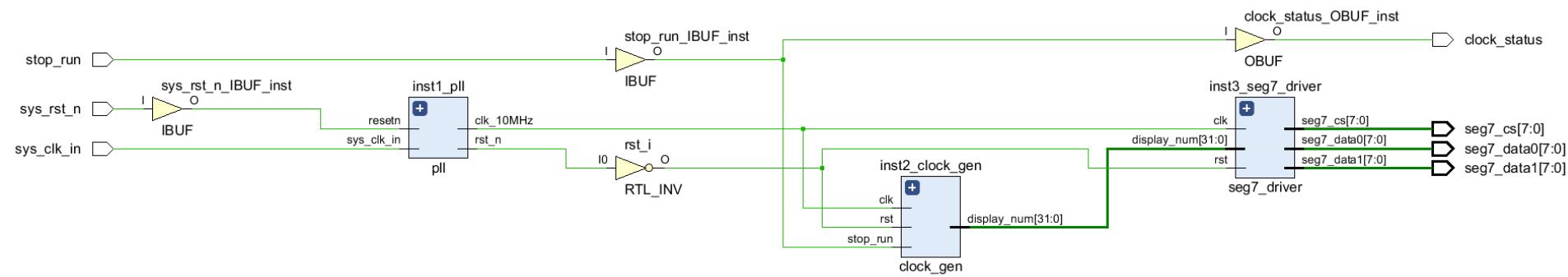


Lab7数码时钟实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

顶层文件(Lab7_top.v)
完成模块互联



Lab7数码时钟实验



```
module clock_gen(
    input clk,
    input rst,
    input stop_run,
    output [31:0] display_num
);
/////////////////////////////
wire ms_marker;
wire sec_marker;
wire min_marker;
wire hour_marker;

/////////////////////////////
time_marker inst1_time_marker
(
    .clk(clk),
    .rst(rst),
    .stop_run(stop_run),
    .ms_marker(ms_marker)
);
```

```
/////////////////////////////
ms_gen inst2_ms_gen
(
    .clk(clk),
    .rst(rst),
    .ms_marker(ms_marker),
    .sec_marker(sec_marker),
    .ms_num(display_num[7:0])
);

/////////////////////////////
sec_gen inst3_sec_gen
(
    .clk(clk),
    .rst(rst),
    .sec_marker(sec_marker),
    .min_marker(min_marker),
    .sec_num(display_num[15:8])
);

/////////////////////////////
min_gen inst4_min_gen
(
    .clk(clk),
    .rst(rst),
    .min_marker(min_marker),
    .hour_marker(hour_marker),
    .min_num(display_num[23:16])
);

/////////////////////////////
hour_gen inst5_hour_gen
(
    .clk(clk),
    .rst(rst),
    .hour_marker(hour_marker),
    .hour_num(display_num[31:24])
);
endmodule
```

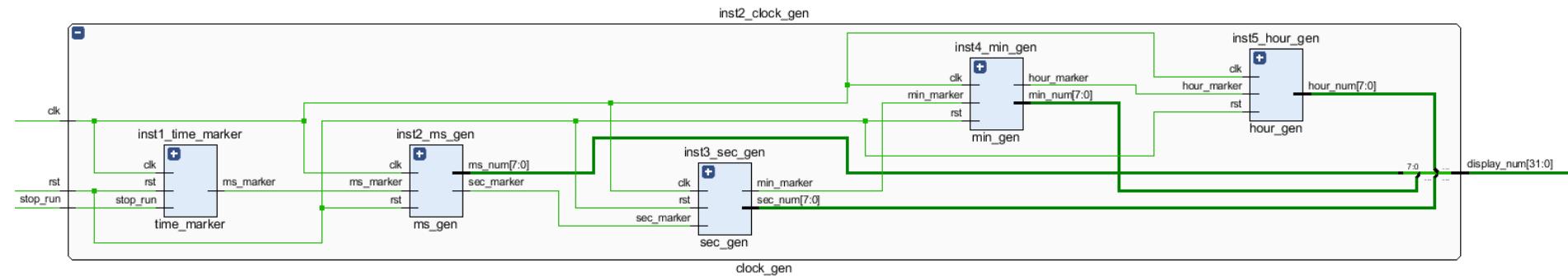
时钟产生模块(clock_gen.v)
产生显示所需的时钟数据

Lab7数码时钟实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

时钟产生模块(**clock_gen.v**)
产生显示所需的时钟数据



Lab7数码时钟实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

```
module time_marker(
    input clk,
    input rst,
    input stop_run,
    output reg ms_marker
);

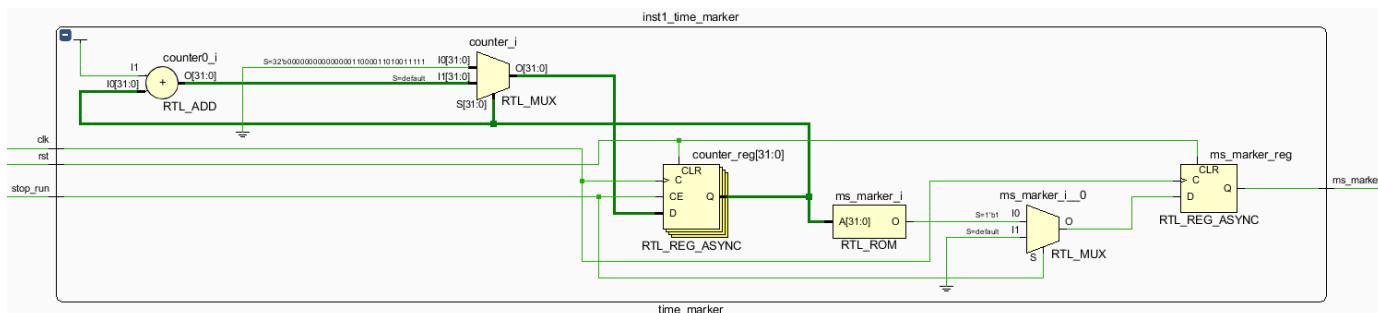
// 定义毫秒时基计数◆?
reg [31:0] counter = 32'd0;

// 定义ms计数器最大值， 10ms/100ns = 100_000;
// 仿真用参数
//parameter COUNTER_10MS = 100;

parameter COUNTER_10MS = 100_000;

//产生10ms为时基的脉冲信号
always @(posedge clk or posedge rst) begin
    if(rst) begin
        counter <= 32'd0;
        ms_marker <= 1'b0;
    end
    else if(stop_run) begin
        if(counter == (COUNTER_10MS-1)) begin
            counter <= 32'd0;
            ms_marker <= 1'b1;
        end
        else begin
            counter <= counter + 1'b1;
            ms_marker <= 1'b0;
        end
    end
    else begin
        counter <= counter;
        ms_marker <= 1'b0;
    end
end
endmodule
```

时基产生模块(time_marker.v)
产生10ms脉冲时基信号



Lab7数码时钟实验

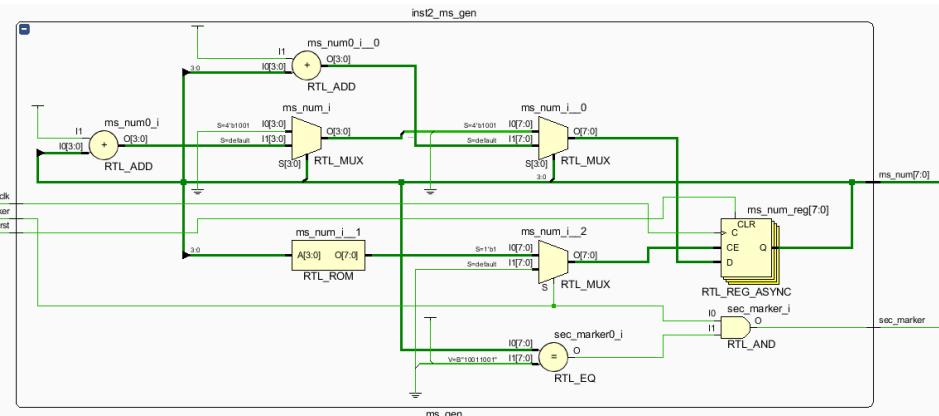


```
module ms_gen(
    input clk,
    input rst,
    input ms_marker,
    output sec_marker,
    output reg [7:0] ms_num
);

    always @(posedge clk or posedge rst) begin
        if(rst) begin
            ms_num <= 8'd0;
        end
        else if(ms_marker) begin
            if(ms_num[3:0] == 4'd9) begin
                ms_num[3:0] <= 4'd0;
                if(ms_num[7:4] == 4'd9) begin
                    ms_num[7:4] <= 4'd0;
                end
                else begin
                    ms_num[7:4] <= ms_num[7:4] + 1'b1;
                end
            end
            else begin
                ms_num[3:0] <= ms_num[3:0] + 1'b1;
            end
        end
        else begin
            ms_num <= ms_num;
        end
    end

    assign sec_marker = ms_marker & (ms_num == 8'h99); //产生秒脉冲标志
endmodule
```

毫秒计数模块(ms_gen.v)
产生毫秒计数值及秒脉冲



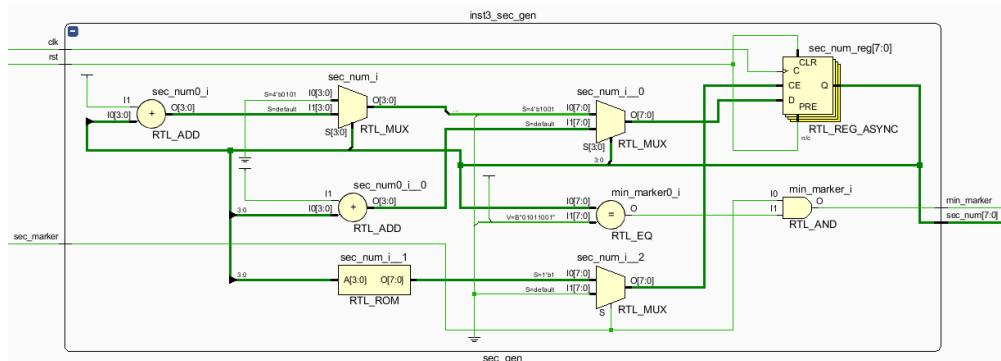
Lab7数码时钟实验



```
module sec_gen(
    input clk,
    input rst,
    input sec_marker,
    output min_marker,
    output reg [7:0] sec_num
);

always @ (posedge clk or posedge rst) begin
    if(rst) begin
        sec_num <= 8'h50;
    end
    else if(sec_marker) begin
        if(sec_num[3:0] == 4'd9) begin
            sec_num[3:0] <= 4'd0; //秒标志到达，个位数为9，则个位清零
            if(sec_num[7:4] == 4'd5) begin
                sec_num[7:4] <= 4'd0; //十位数为5，则十位清零
            end
            else begin
                sec_num[7:4] <= sec_num[7:4] + 1'b1; //十位不是5，则十位+1
            end
        end
        else begin
            sec_num[3:0] <= sec_num[3:0] + 1'b1; //个位不是9，则个位+1
        end
    end
    else begin
        sec_num <= sec_num; //秒脉冲未到，则保持
    end
    assign min_marker = sec_marker & (sec_num == 8'h59); //输出分钟脉冲
endmodule
```

秒计数模块(sec_gen.v)
产生秒计数值及分钟脉冲



Lab7数码时钟实验

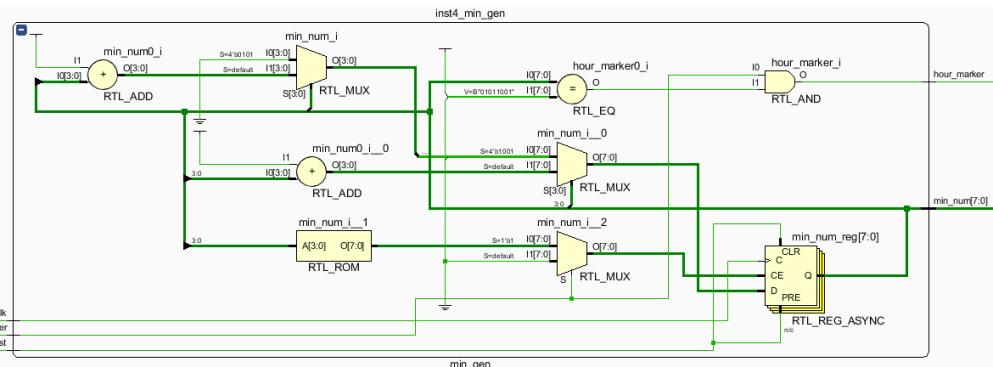


```
module min_gen(
    input clk,
    input rst,
    input min_marker,
    output hour_marker,
    output reg [7:0] min_num
);

always @(posedge clk or posedge rst) begin
    if(rst) begin
        min_num <= 8'h59;
    end
    else if(min_marker) begin
        if(min_num[3:0] == 4'd9) begin
            min_num[3:0] <= 4'd0;
            if(min_num[7:4] == 4'd5) begin
                min_num[7:4] <= 4'd0;
            end
            else begin
                min_num[7:4] <= min_num[7:4] + 1'b1;
            end
        end
        else begin
            min_num[3:0] <= min_num[3:0] + 1'b1;
        end
    end
    else begin
        min_num <= min_num;
    end
    end

    assign hour_marker = min_marker & (min_num == 8'h59); //产生小时脉冲
endmodule
```

分钟计数模块(min_gen.v)
产生分钟计数值及小时脉冲



Lab7数码时钟实验

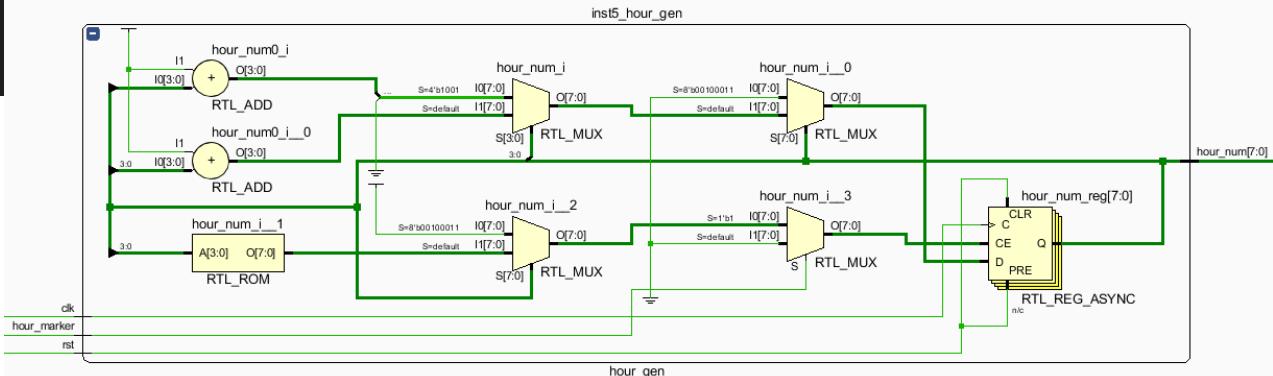


國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

```
module hour_gen(
    input clk,
    input rst,
    input hour_marker,
    output reg [7:0] hour_num
);

    always @ (posedge clk or posedge rst) begin
        if(rst) begin
            hour_num <= 8'h23;
        end
        else if(hour_marker) begin
            if(hour_num == 8'h23) begin
                //小时标志到达
                //是否为23时，是则清零
                hour_num <= 8'd0;
            end
            else if(hour_num[3:0] == 4'd9) begin
                //不是23时，个位是否为9，是则个位清零，十位+1
                hour_num[3:0] <= 4'd0;
                hour_num[7:4] <= hour_num[7:4] + 1'b1;
            end
            else begin
                //不是23时，个位不是9，则个位+1
                hour_num[3:0] <= hour_num[3:0] + 1'b1;
            end
        end
        else begin
            //小时脉冲未到，则保持
            hour_num <= hour_num;
        end
    end
endmodule
```

小时计数模块(hour_gen.v)
产生小时计数值



Lab7数码时钟实验



```
// 定义数码管刷新周期计数器
reg [31:0] cnt_refresh;
// 定义当前正常刷新的数码管编号
reg [3:0] led_num = 4'd0;

// 刷新计数间隔设置为1ms, 1ms/100ns = 10_000
parameter REFRESH_TIME = 10_000;

// 定义当前显示数据
reg [3:0] current_display_num;

// 产生数码管刷新显示计数器, 每1ms切换一位
always @(posedge clk_10MHz or posedge rst) begin
    if(rst) begin
        cnt_refresh <= 32'd0;
        led_num <= 3'd0;
    end
    else if(cnt_refresh == (REFRESH_TIME-1)) begin
        cnt_refresh <= 32'd0;
        if(led_num == 4'd7) begin
            led_num <= 4'd0;
        end
        else begin      生成数码管刷新计数
            led_num <= led_num + 1'b1;
        end
    end
    else begin
        cnt_refresh <= cnt_refresh + 1'b1;
    end
end
```

```
//根据数码管刷新显示计数器, 选择输出当前的字段与位选信号
always @(posedge clk_10MHz or posedge rst) begin
    if(rst) begin
        current_display_num <= NUM0;
        seg7_cs <= CSN;
    end
    else begin      根据数码管刷新计数值  
送出对应的字段与位选信号
        case(led_num)
            4'd0: begin
                current_display_num <= display_num[3:0];
                seg7_cs = CS0;
            end
            4'd1: begin
                current_display_num <= display_num[7:4];
                seg7_cs <= CS1;
            end
            4'd2: begin
                current_display_num <= display_num[11:8];
                seg7_cs <= CS2;
            end
            4'd3: begin
                current_display_num <= display_num[15:12];
                seg7_cs <= CS3;
            end
            4'd4: begin
                current_display_num <= display_num[19:16];
                seg7_cs <= CS4;
            end
            4'd5: begin
                current_display_num <= display_num[23:20];
                seg7_cs <= CS5;
            end
            4'd6: begin
                current_display_num <= display_num[27:24];
                seg7_cs <= CS6;
            end
            4'd7: begin
                current_display_num <= display_num[31:28];
                seg7_cs <= CS7;
            end
            default: begin
                current_display_num <= NUM0;
                seg7_cs <= CSN;
            end
        endcase
    end
end
```

数码管驱动模块(seg7_driver.v)
通过字库索引输出数据

```
//根据数码管显示数据进行字库索引
always @(posedge clk_10MHz or posedge rst) begin
    if(rst) begin
        seg7_data0 <= NUM0;
    end
    else begin      字库索引输出
        case(current_display_num)
            4'h0: seg7_data0 <= NUM0;
            4'h1: seg7_data0 <= NUM1;
            4'h2: seg7_data0 <= NUM2;
            4'h3: seg7_data0 <= NUM3;
            4'h4: seg7_data0 <= NUM4;
            4'h5: seg7_data0 <= NUM5;
            4'h6: seg7_data0 <= NUM6;
            4'h7: seg7_data0 <= NUM7;
            4'h8: seg7_data0 <= NUM8;
            4'h9: seg7_data0 <= NUM9;
            4'ha: seg7_data0 <= NUMA;
            4'hb: seg7_data0 <= NUMB;
            4'hc: seg7_data0 <= NUMC;
            4'hd: seg7_data0 <= NUMD;
            4'he: seg7_data0 <= NUME;
            4'hf: seg7_data0 <= NUMF;
            default: seg7_data0 <= NUM0;
        endcase
    end
    seg7_data1 <= seg7_data0;
end
```



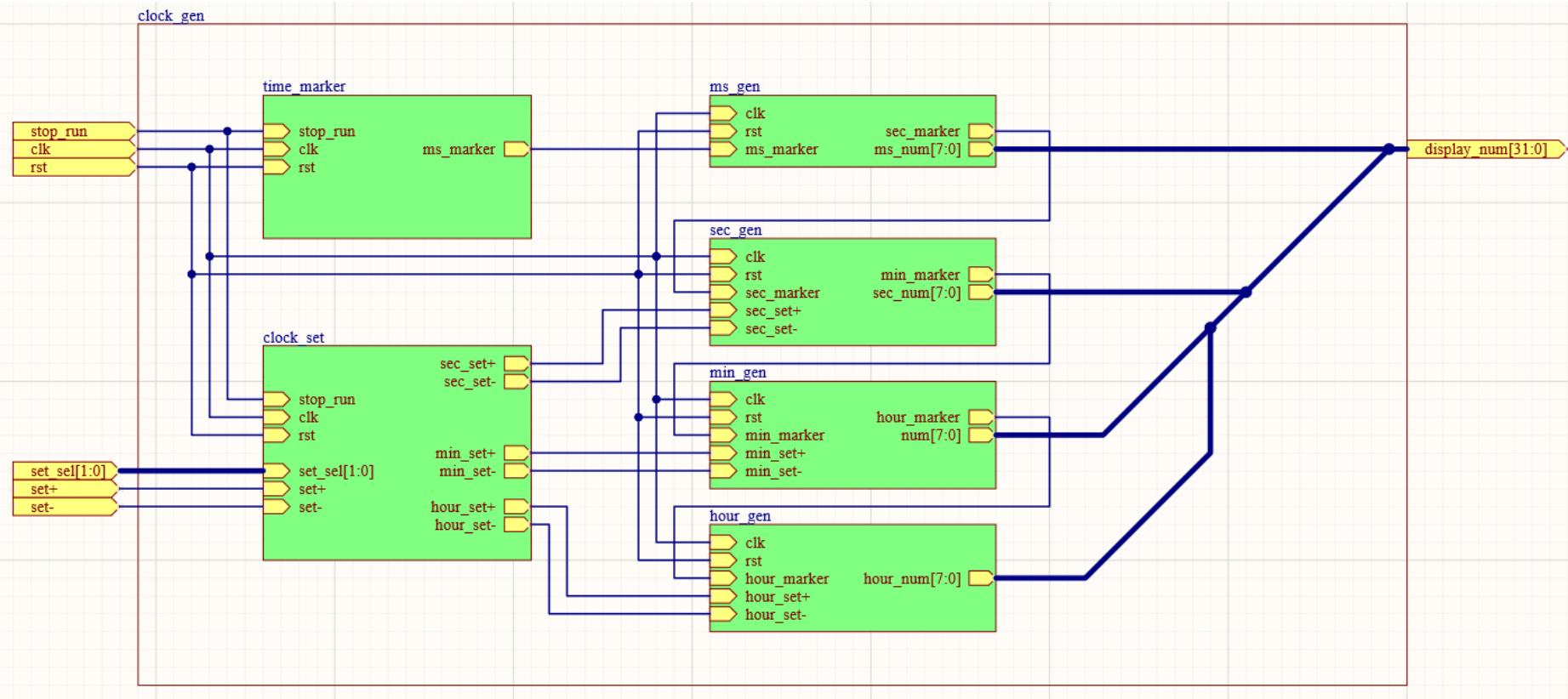
效果2参考思路

1. stop_run作为时钟设置信号， stop状态下可以对时钟进行设置
2. clock_gen模块内部增加时钟设置模块
3. 使用拨码开关切换要设置的位置 (00-秒, 01-分, 10-时)
4. 使用按键对时间进行调整[S4-增加(set+), S5-减小(set-)]
5. 注意合理设计set+/set-的控制时序
6. 体会按键抖动带来的影响

Lab7数码时钟实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS



下节预告



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Lab8实验内容

Lab8 XADC实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 实验内容

- 通过XADC采集EGo1电路板上电位计的模拟电压
- 将采集到的模拟电压显示在数码管上
- 实验1：显示电位计的模拟电压
- 实验2：通过按键切换显示电位计模拟电压及片上温度

■ 实验目的

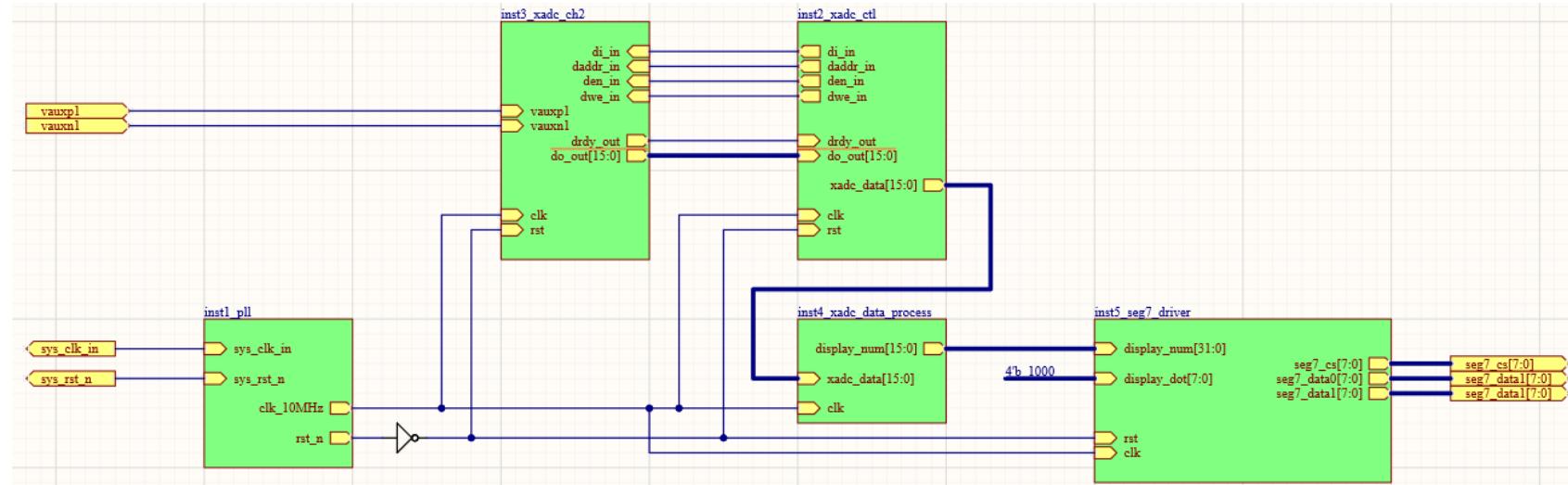
- 熟悉XADC的使用方法
- 学会ILA的使用
- 学会计算类（乘法及除法）IP核的使用



Lab8 XADC实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



■ 系统框图

- 电路共包括五个模块
- pll模块完成系统时钟管理，通过IP管理器配置、例化后实现
- xadc_ch2模块为2通道AD采集模块，通过IP管理器配置、例化后实现
- xadc_ctl模块为xadc_ch2时序控制模块，产生xadc工作所需的时序信号，采样得到xadc_data数据
- xadc_data_process模块为数据处理模块，完成二进制数据至BCD码转换
- seg7_driver模块为数码管驱动模块，完成xadc数据显示

Lab8 XADC实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

输入输出端口表

序号	信号名称	信号类型	输入/输出	信号描述	FPGA引脚	说明
1	sys_clk_in	1bit	输入	系统时钟输入	P17	100MHz输入
2	sys_rst_n	1bit	输入	系统复位输入	P15	低电平复位
3	switch_display	1bit	输入	数据切换输入	R1	高电平使能有效
4	switch_status	1bit	输出	数据状态显示	K2	高电平点亮
5	seg7_cs[7:0]	8bit	输出	数码管位选控制信号输出	G2/C2/C1/H1 G1/F1/E1/G6	高电平选通
6	seg7_data0[7:0]	8bit	输出	数码管字段控制信号输出	B4/A4/A3/B1 A1/B3/B2/D5	高电平有效
7	seg7_data1[7:0]	8bit	输出	数码管字段控制信号输出	D4/E3/D3/F4 F3/E2/D2/H2	高电平有效
8	vauxp1	1bit	输入	模拟电压输入	C12	电位计模拟信号输入
9	vauxn1	1bit	输入	模拟电压输入	B12	GND
10	vp_in	1bit	输入	模拟电压输入	J10	空载
11	vn_in	1bit	输入	模拟电压输入	K9	空载



THANKS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计

第十讲 (10.1 XADC概述及使用)

2025年5月6日

本节内容



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

- 1. 7系列XADC介绍**
- 2. 7系列XADC使用方法**
- 3. Vivado中ILA的使用**



1. 7系列XADC介绍

7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter

User Guide

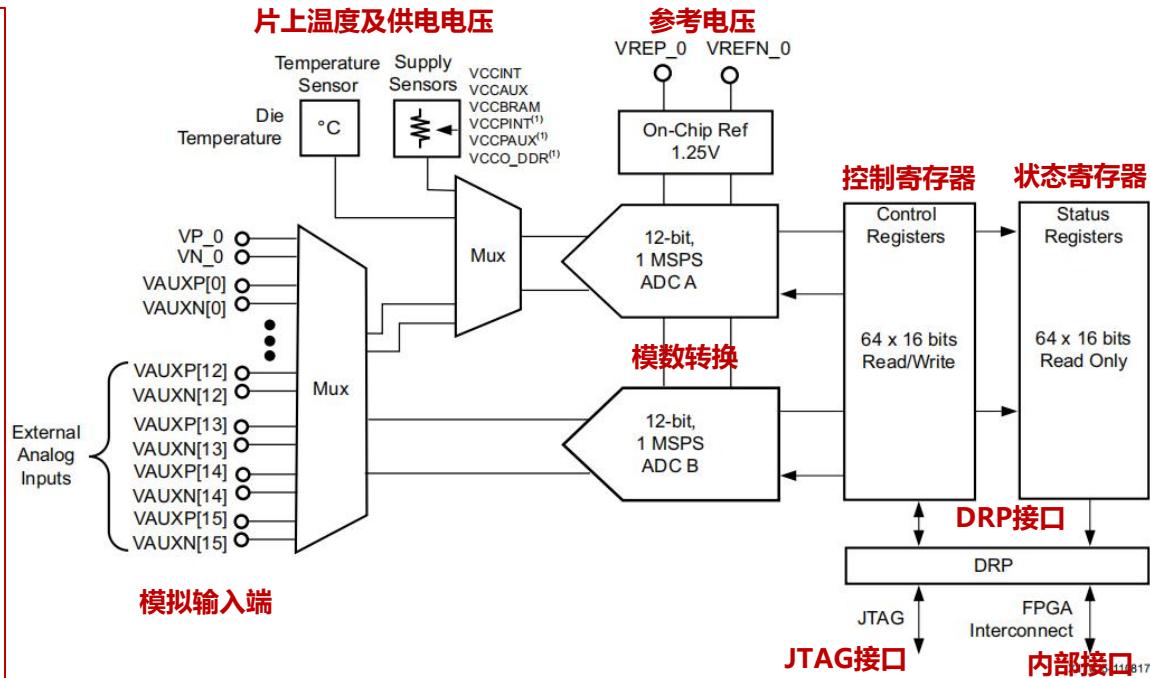
7系列XADC介绍



XADC, Xilinx Analog-to-Digital Converter

■ XADC的技术指标

- 分辨率:** 两路(ADC A & ADC B)12bit
- 转换速率:** $\leq 1\text{MSPS}$
- 采集模式:** 连续模式, 事件(外触发)模式
- 通道模式:** 同时转换模式, 独立模式, 单通道模式, 通道序列模式, 外部多路复用模式
- 输入通道:** 芯片温度, 芯片供电电压, 专用模拟端口(VP,VN), 复用模拟端口(VAUXP,VAUXN)
- 参考电压:** 片上1.25V或片外供电
- 模拟输入电压范围:** 单极性(0~1V), 双极性($\pm 0.5\text{V}$)

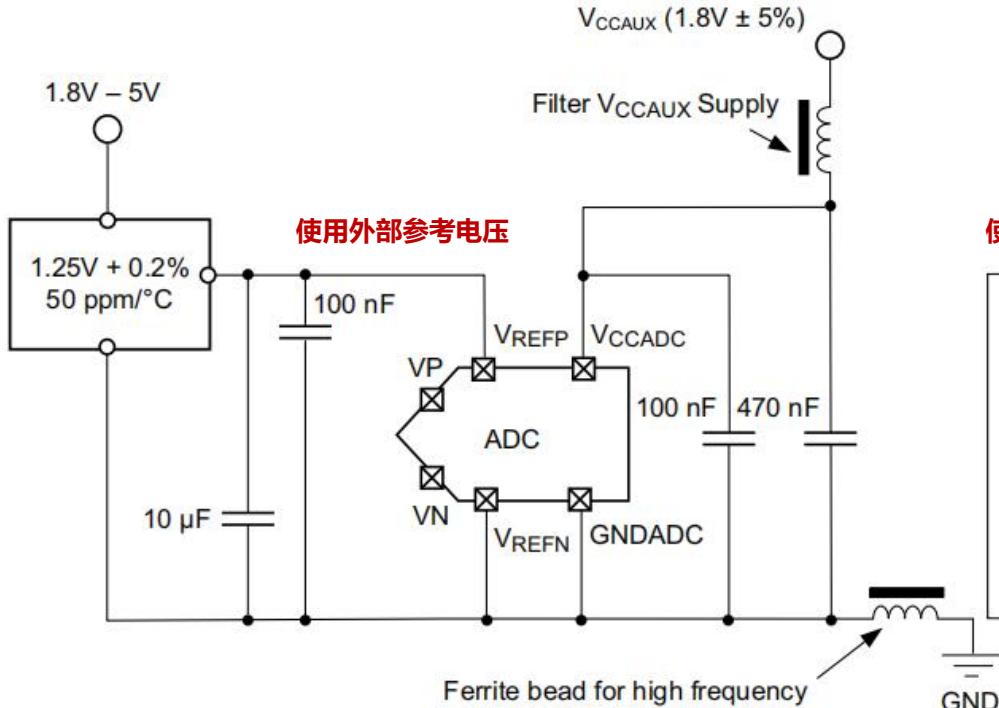


7系列XADC介绍

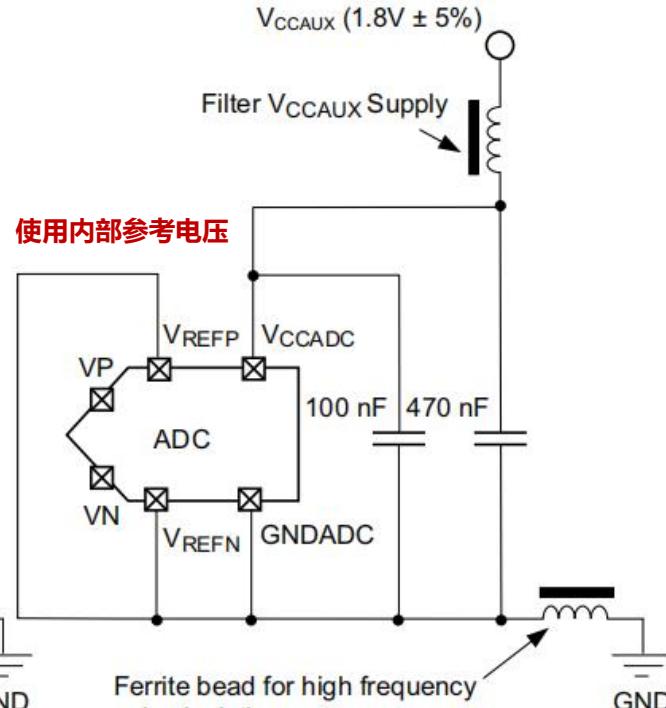


國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Use External Reference IC



Enable On-Chip Reference



① 参考电压

☒ Package Pins

7系列XADC介绍



■ 单极性输入

- 电压输入范围: $0V \leq V_P - V_N \leq 1V$
- V_N 通常接地
- 允许 V_N 变化的范围为 $0 \sim +0.5V$
- V_P 允许输入的最大值为 $+1.5V$

■ 双极性输入

- 电压输入范围: $-0.5V \leq V_P - V_N \leq +0.5V$
- V_N 接地, V_P 输入的范围为 $\pm 0.5V$
- V_N 允许的最大值为 $0.5V$

② 模拟电压输入

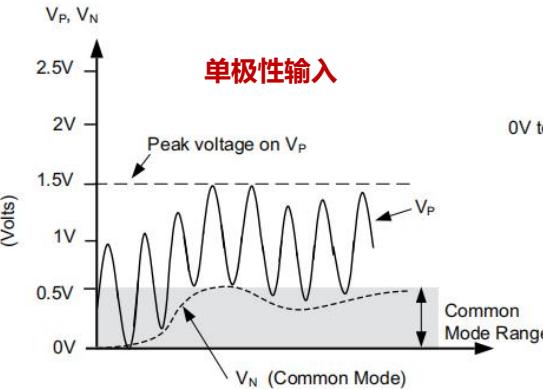


Figure 2-6: Unipolar Input Signal Range

X17022-110817

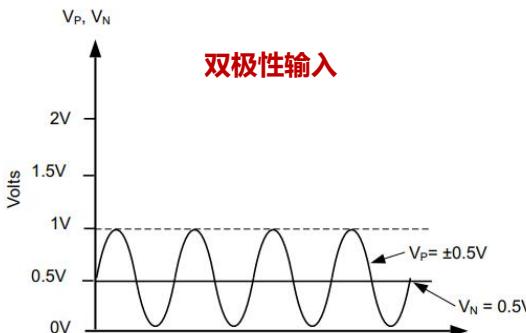
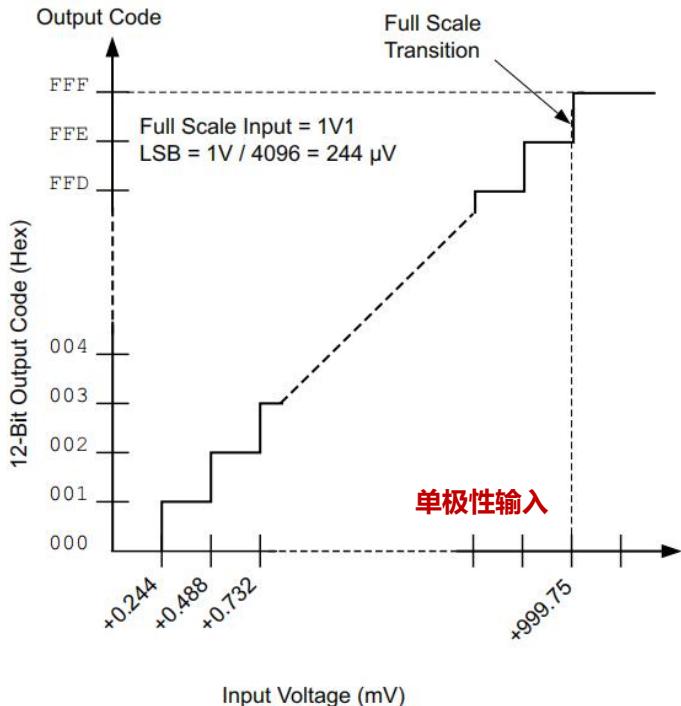


Figure 2-7: Bipolar Input Signal Range

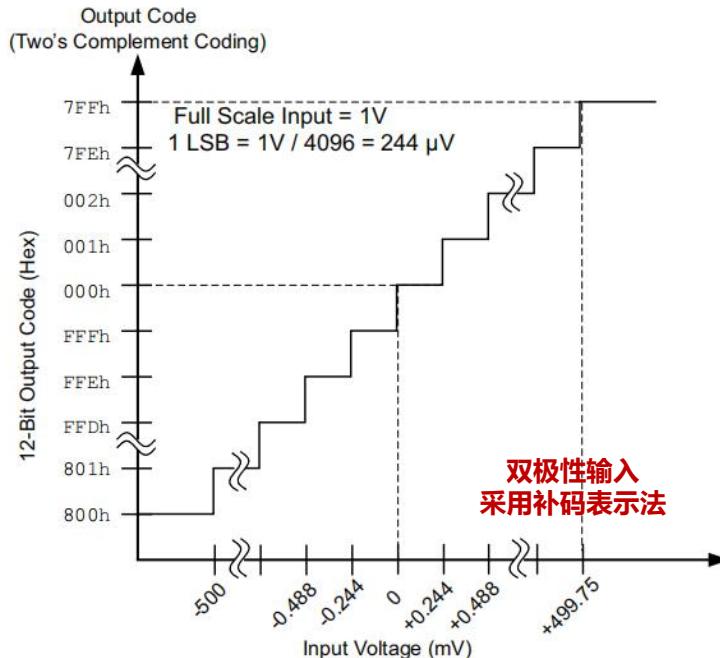
X17023-110817

7系列XADC介绍



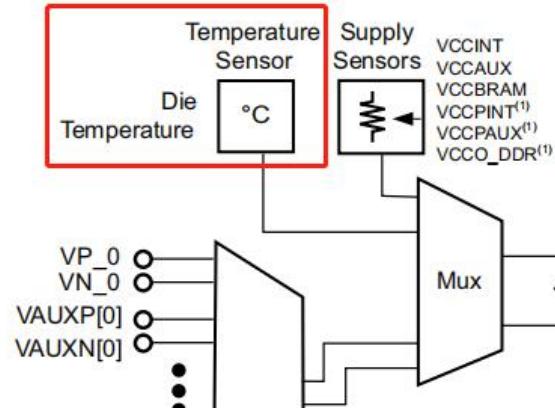
$$Voltage(mv) = \frac{ADC\ code \times 1000}{4096}$$

② 模拟电压输入



$$Voltage(mv) = \frac{ADC\ code \times 1000}{4096}$$

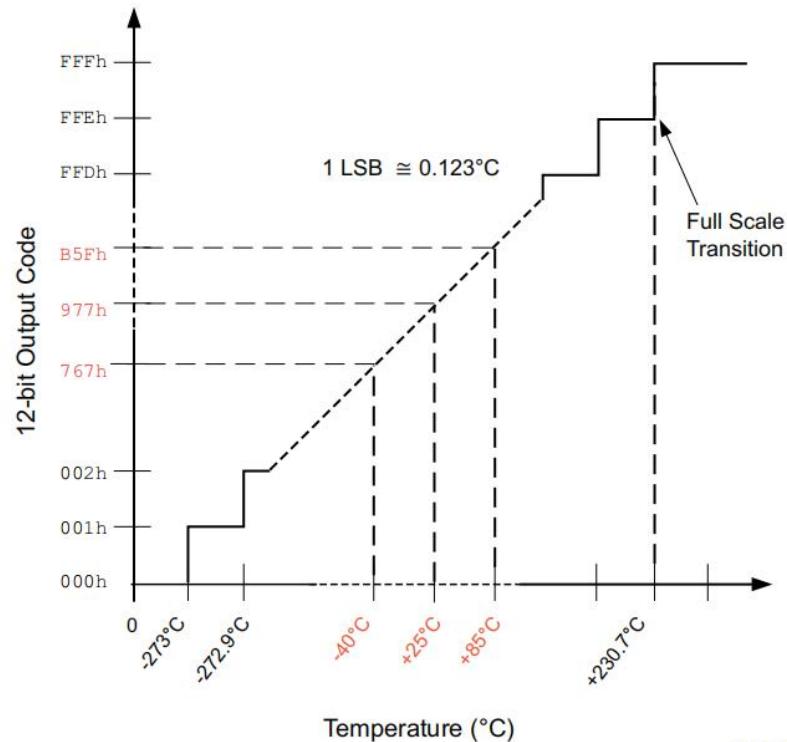
7系列XADC介绍



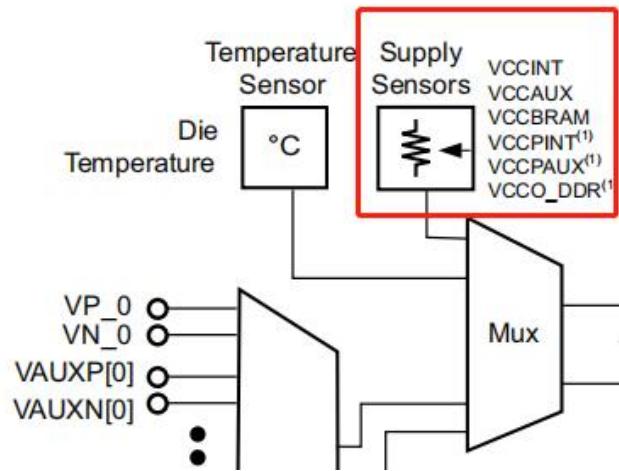
③ 温度传感器输入

$$\text{Temperature } (\text{°C}) = \frac{\text{ADC Code} \times 503.975}{4096} - 273.15$$

Equation 2-6

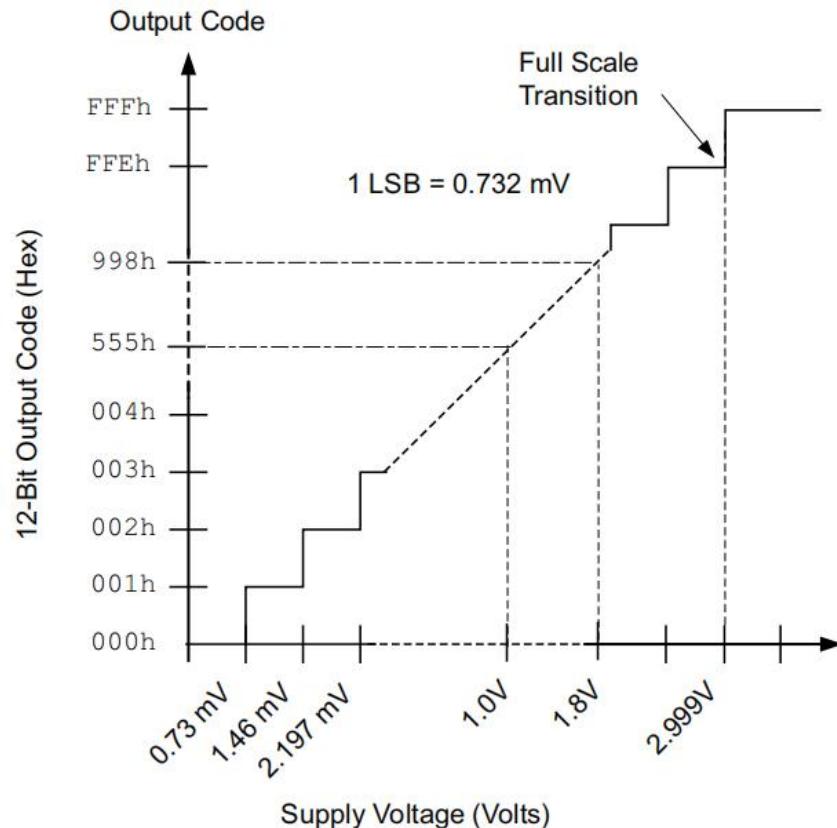


7系列XADC介绍



$$\text{Supply Voltage}(V) = \frac{\text{ADC Code} \times 3V}{4096}$$

④ 供电电压输入



7系列XADC介绍



- **DRP:** 动态重配置接口
- **控制和时钟:** 复位(RESET)
事件模式下的转换控制(CONVST、CONVSTCLK)
- **外部模拟输入:** VP,VN,VAUXP,VAUXN
- **报警输出:** ALM[7:0], OT
- **状态输出:** JTAG控制接口

ADC转换状态 (EOC,EOS,BUSY)
通道标记 (MUXADDR,CHANNEL)

ALM[0] ⁽²⁾	Output	Temperature sensor alarm output.
ALM[1] ⁽²⁾	Output	V _{CCINT} sensor alarm output.
ALM[2] ⁽²⁾	Output	V _{CCAUX} sensor alarm output.
ALM[3] ⁽²⁾	Output	V _{CCBRAM} sensor alarm output.
ALM[4] ⁽⁴⁾	Output	V _{CCPINT} sensor alarm output.
ALM[5] ⁽⁴⁾	Output	V _{CCPAUX} sensor alarm output.
ALM[6] ⁽⁴⁾	Output	V _{CCO_DDR} sensor alarm output.
ALM[7] ⁽²⁾	Output	Logic OR of bus ALM[6:0]. Can be used to flag the occurrence of any alarm.
OT ⁽²⁾	Output	Over-Temperature alarm output.

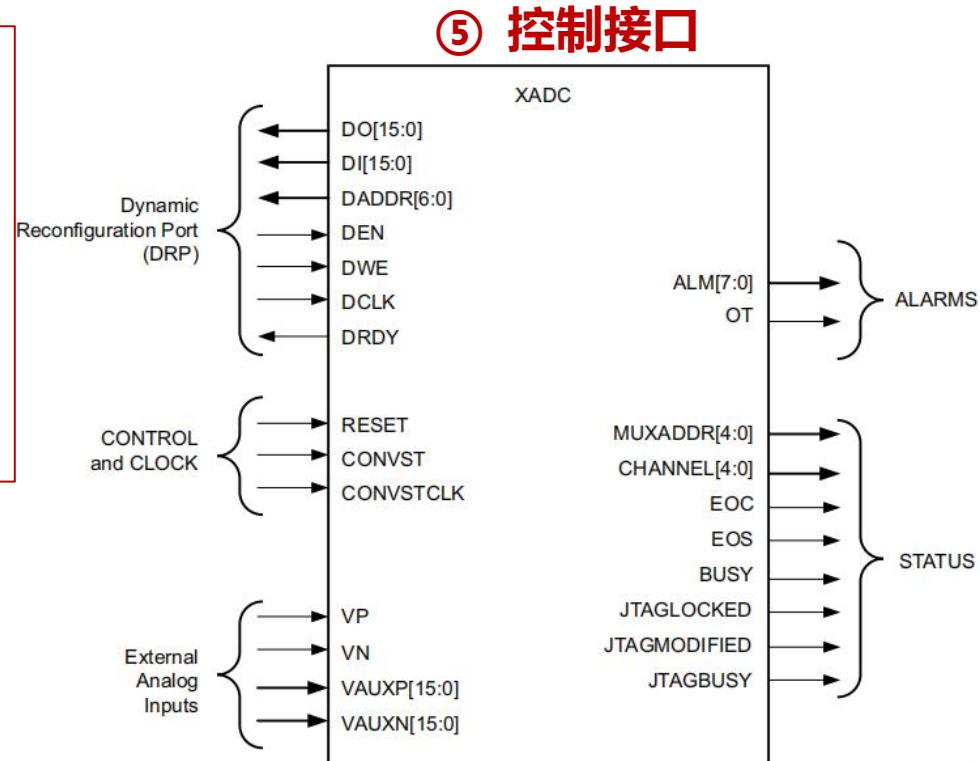


Figure 1-3: XADC Primitive Ports

7系列XADC介绍



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

⑥ 内部寄存器

■ 状态寄存器（只读）

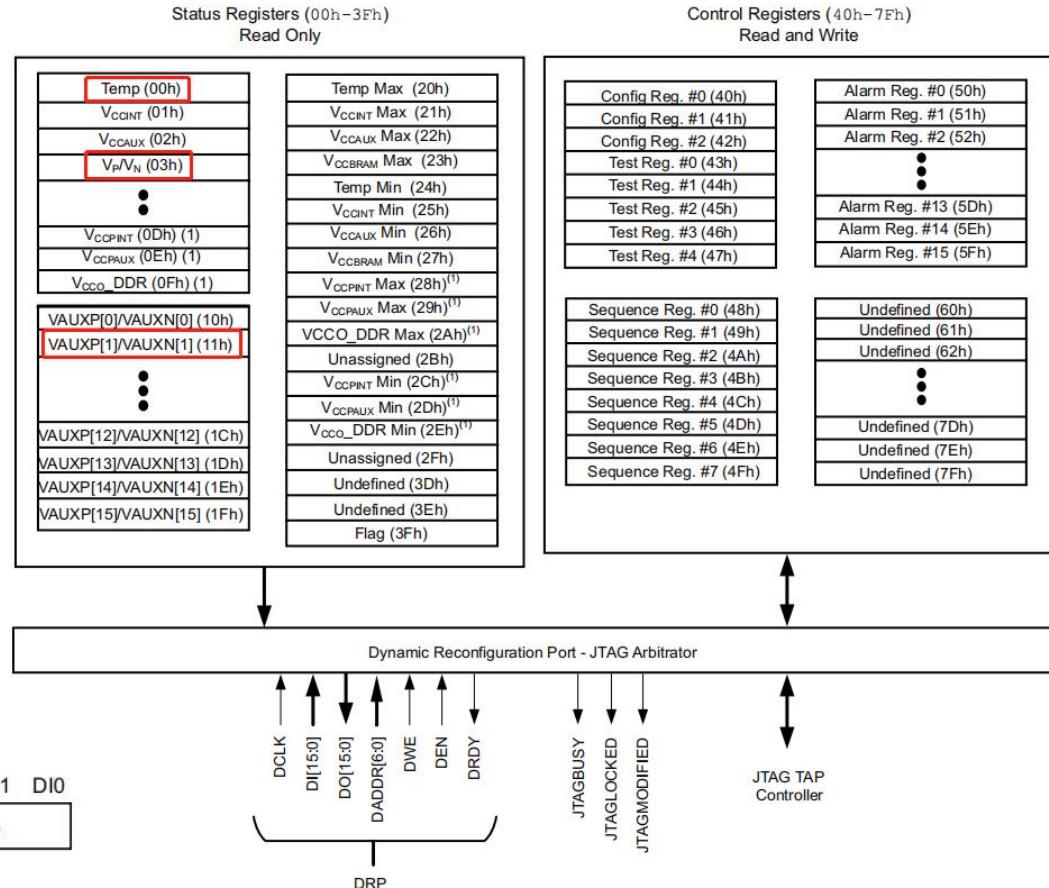
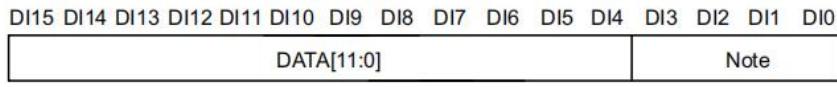
- 地址范围: 00h~3Fh
- 存储转换结果数据

■ 控制寄存器（读/写）

- 地址范围: 40h~7Fh
- XADC控制数据

■ 补充说明

- 通过读取状态寄存器的数据，得到转换结果
- 通过写入控制寄存器，实现XADC配置
- **XADC转换结果存储在高12位！！！**

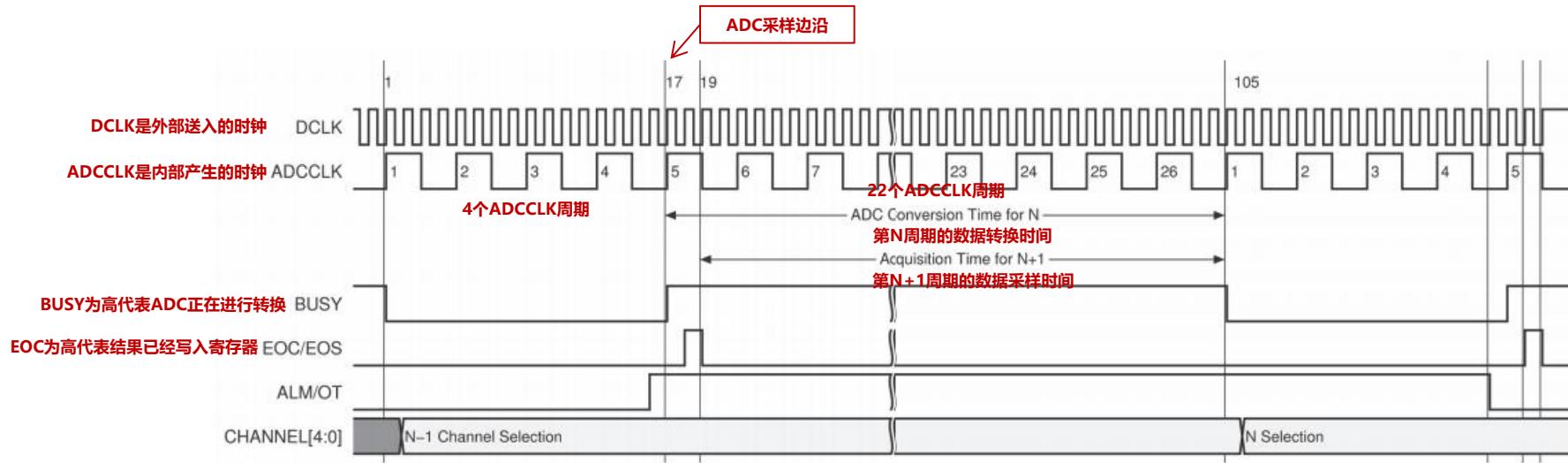


7系列XADC介绍



⑦ 连续采样工作模式

1. 通过DCLK产生ADCCLK
2. XADC根据设置的通道进行连续转换
3. 结果存入状态寄存器



X17038-110817

Figure 5-1: Continuous Sampling Mode

7系列XADC介绍



⑧ 事件（外触发）工作模式

1. 通过DCLK产生ADCCLK
2. XADC在CONVST信号的上升沿启动转换
3. 结果存入状态寄存器

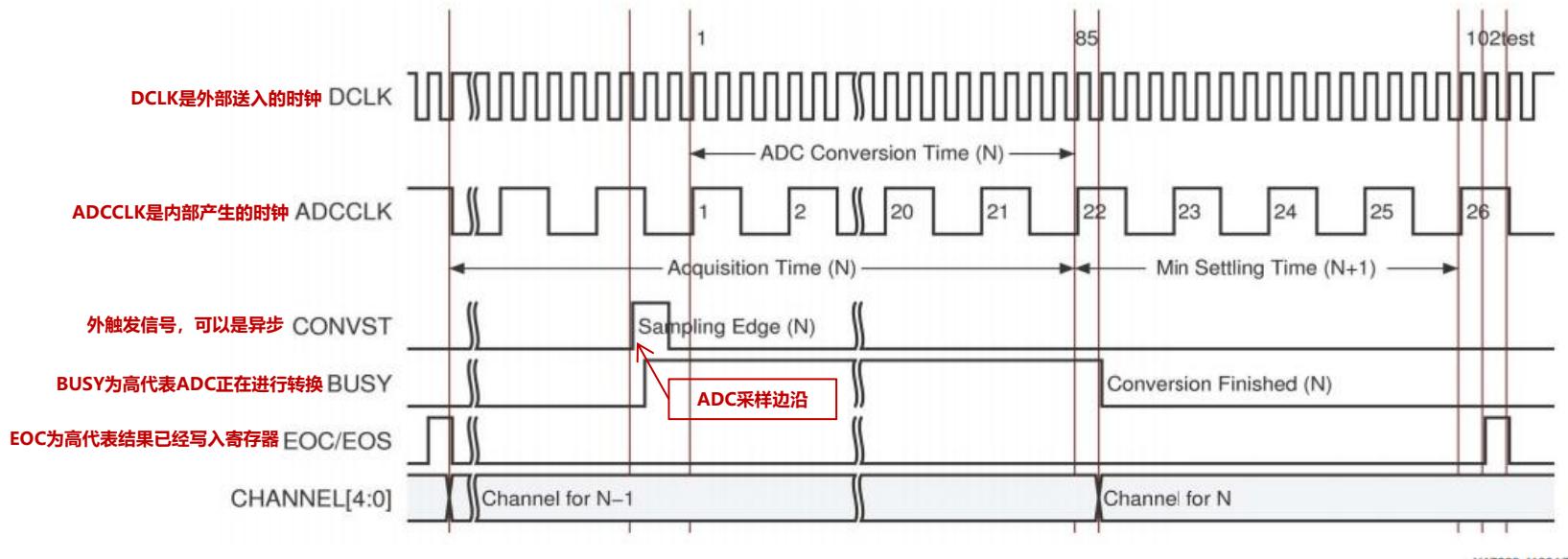


Figure 5-2: Event Driven Sampling Mode

7系列XADC介绍



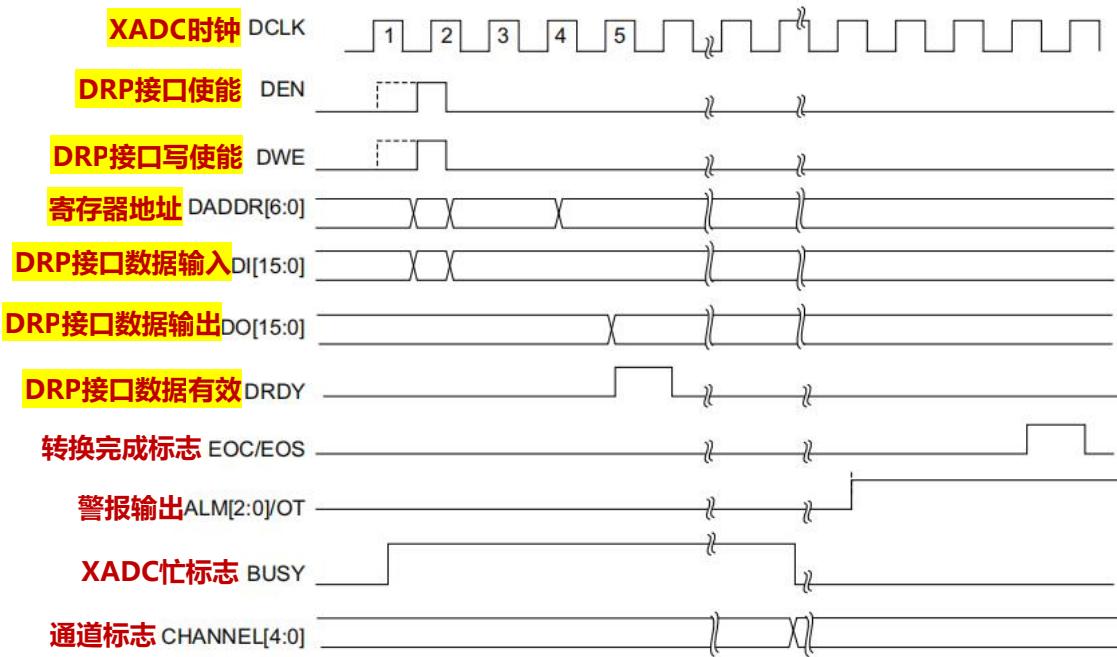
■ 读状态寄存器

- 目的是读取转换结果
- 使用的信号线：
DCLK, DEN, DWE, DADDR, DO, DRDY
- 控制时序：
 - (1) DEN置高一个clk周期，保持DWE低电平（进入DRP读模式），同时送出需要读取的寄存器地址(DADDR)；
 - (2) 等待DRDY信号变高，变高则读取DO数据。

■ 读写控制寄存器

- 目的是设置或读取控制寄存器
- 使用的信号线：
DCLK, DEN, DWE, DADDR, DI, DO, DRDY
- 控制时序（写入）：
 - (1) DEN, DWE置高一个clk周期,同时送出需要写入的寄存器地址(DADDR)及数据 (DI)；
 - (2) 等待DRDY信号变高，变高则数据写入成功。
- 控制时序（读取）：
与状态寄存器读取时序一致

⑨ 控制接口工作时序



X17040-110817

Figure 5-3: DRP Detailed Timing

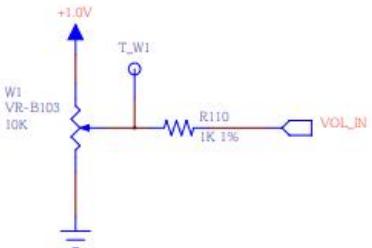


2. 7系列XADC使用方法

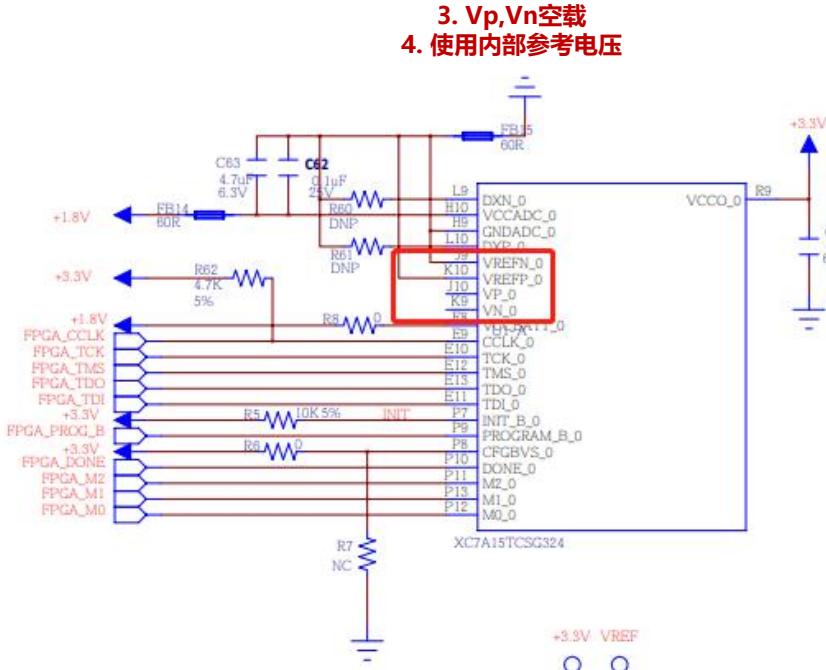
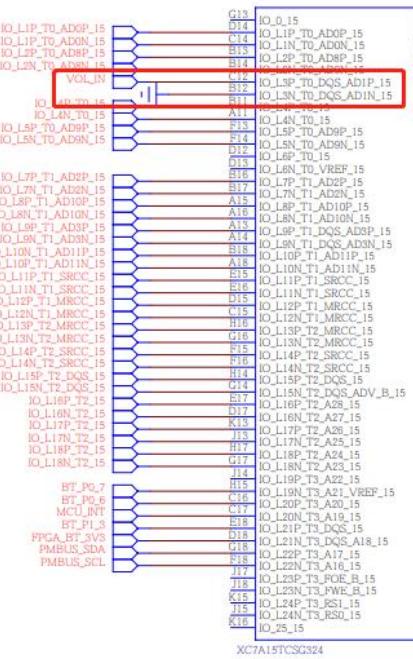


7系列XADC使用方法

① EGo1 XADC相关硬件电路



2. 接入VAUXP[1], 单极性模式





② 配置XADC IP核

通过例化XADC IP核实现控制及数据读取

The screenshot shows the Vivado IP Catalog interface. The search bar at the top contains "XADC" and displays "(2 matches)". The results table has columns for Name, Status, License, and VLVN. Under the "Name" column, there are three entries: "AXI4", "Status", and "License". The "Vivado Repository" section shows "FPGA Features and Design" expanded, with "XADC" further expanded, and the "XADC Wizard" item highlighted with a red box. Below the table, the status bar shows "AXI4, AXI4-Stream Production Included xilinx.com:ip:xadc_wiz:3.3".

7系列XADC使用方法



Show disabled ports

Component Name: xadc_ch2

1. IP名称

Basic ADC Setup Alarms Channel Sequencer Summary

2. 接口模式-DRP

AXI4Lite DRP None

3. 时序模式-连续采集

Continuous Mode Event Mode

4. 通道模式-同时采集

Simultaneous Selection Independent ADC Single Channel Channel Sequencer

5. IP主时钟

Enable DCLK DCLK Frequency(MHz) 10 [8.0 - 250.0]

ADC Conversion Rate(KSPS) 100 [39.0 - 193.0]

6. XADC采样率

Acquisition Time (CLK) 4

Clock divider value = 4

ADC Clock Frequency(MHz) = 2.50

7. 复位信号

reset_in Temp us Temp register

8. 仿真数据文件 (可选)

Analog Sim File Options

Sim File Selection	Default
Analog Stimulus File	design
Sim File Location	/
Waveform Type	CONSTANT
Frequency (KHz)	1.0 [0.1 - 480.77]
Number of Wave	1 [1 - 1000]

7系列XADC使用方法



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

② 配置XADC IP核

Component Name: xadc_ch2

Basic ADC Setup Alarms Channel Sequencer Summary

Sequencer Mode: Off

ADC Calibration

ADC Offset Calibration

ADC Offset and Gain Calibration

Enable CALIBRATION Averaging

External Multiplexer Setup

External Multiplexer

Channel for MUX: VP VN

Enable muxaddr_out port

Power Down Options

ADCB

ADCA

Channel Averaging: None

Supply Sensor Calib:

- None
- 16
- 64
- 256

可以设置输入通道的平均点数

Component Name: xadc_ch2

Basic ADC Setup Alarms Channel Sequencer Summary

Over Temperature Alarm (°C)

Trigger: 125.0 [-40.0 - 125.0]

Reset: 70.0 [-40.0 - 125.0]

User Temperature Alarm (°C)

Trigger: 85.0 [-40.0 - 125.0]

Reset: 60.0 [-40.0 - 125.0]

VCCINT Alarm (Volts)

Lower: 0.97 [0.0 - 1.05]

Upper: 1.03 [0.0 - 1.05]

VCCAUX Alarm (Volts)

Lower: 1.75 [0.0 - 1.89]

Upper: 1.89 [0.0 - 1.89]

VCCBRAM Alarm (Volts)

Lower: 0.95 [0.0 - 1.05]

Upper: 1.05 [0.0 - 1.05]

可以设置报警输出及报警范围

7系列XADC使用方法



② 配置XADC IP核

Component Name: xadc_ch2

	Channel Enable	Average Enable	Bipolar	Acquisition Time
TEMPERATURE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
VCCINT	<input type="checkbox"/>	<input type="checkbox"/>		
VCCAUX	<input type="checkbox"/>	<input type="checkbox"/>		
VCCBRAM	<input type="checkbox"/>	<input type="checkbox"/>		
VP/VN	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
VREFP	<input type="checkbox"/>			
VREFN	<input type="checkbox"/>			
vauxp0/vauxn0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
vauxp1/vauxn1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
vauxp2/vauxn2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

选择需要的输入通道
可以勾选平均选项，降低噪声

7系列XADC使用方法



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  
xadc_ch2 inst3_xadc_ch2  
(  
    .di_in(di_in),                                // input wire [15 : 0] di_in  
    .daddr_in(daddr_in),                            // input wire [6 : 0] daddr_in  
    .den_in(den_in),                               // input wire den_in  
    .dwe_in(dwe_in),                               // input wire dwe_in  
    .drdy_out(drdy_out),                            // output wire drdy_out  
    .do_out(do_out),                               // output wire [15 : 0] do_out  
    .dclk_in(clk_10MHz),                           // input wire clk  
    .reset_in(rst),                                // input wire rst  
    .vp_in(vp_in),                                 // input wire vp_in  
    .vn_in(vn_in),                                 // input wire vn_in  
    .vauxp1(vauxp1),                               // input wire vauxp1  
    .vauxn1(vauxn1),                               // input wire vauxn1  
    .vauxp9(),                                    // input wire vauxp9  
    .vauxn9(),                                    // input wire vauxn9  
    .channel_out(),                                // output wire [4 : 0] channel_out  
    .eoc_out(),                                    // output wire eoc_out  
    .alarm_out(),                                  // output wire alarm_out  
    .eos_out(),                                    // output wire eos_out  
    .busy_out()                                    // output wire busy_out  
);  
不使用的信号可以不连接
```

③ 例化XADC IP核

■ 例化XADC IP核

- 根据选定的模式给出控制信号
- 不使用的信号可以不连接

■ XADC IP核控制

- 根据控制时序设计XADC控制模块
- 多路信号采集时可以采用状态机



3. Vivado中ILA的使用

Vivado中ILA的使用

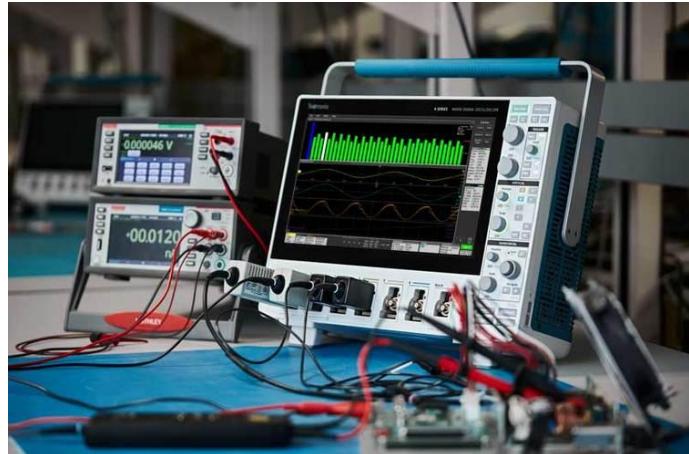


國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

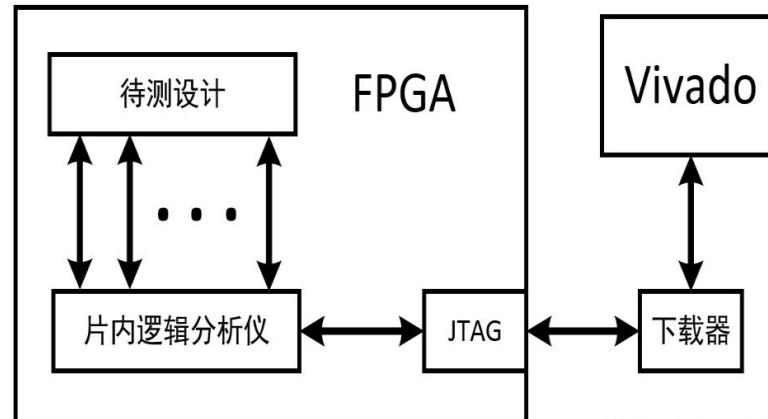
■ ILA功能介绍

- 由于FPGA综合、实现的电路在芯片内部，是没有办法使用示波器或逻辑分析仪测量内部信号的，所以Xilinx等厂家发明了内置逻辑分析仪，在Vivado软件中取名为ILA (Integrated Logic Analyzer)，在之前的ISE软件中取名为ChipScope，原理是使用FPGA内部的门电路搭建逻辑分析仪，在芯片内部接出probe去探测信号线上的信号
- 相比于仿真，ILA能够监测FPGA内部的真实信号，可以更好地定位电路问题，但ILA本身需要占用FPGA内部资源，更改原有的布局布线，可能对原有的电路产生一定的影响

板级信号测量



FPGA 内部信号测量



Vivado中ILA的使用



■ ILA使用方式

- 方式一：调用ILA的IP核并例化

优点：层次化设计，结构清晰

缺点：灵活性差，不能测量全部信号，不使用时需要修改HDL代码

- 方式二：使用Set Up Debug的方式

优点：灵活性好，测量信号多，不使用时不需要修改HDL代码

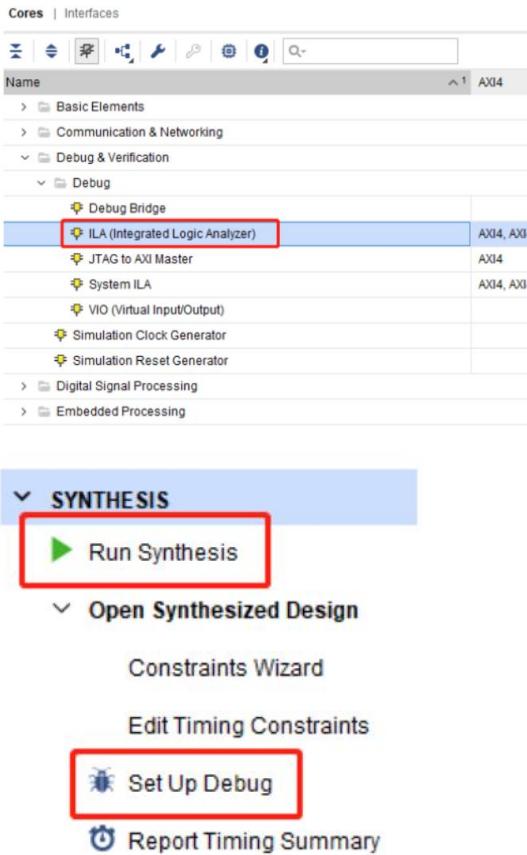
缺点：信号量多时较为混乱

- 方式三：XDC 约束文件中书写对应的Tcl XDC 调试命令

优点：纯文本方式

缺点：不直观

https://blog.csdn.net/qq_33155311/article/details/107839056



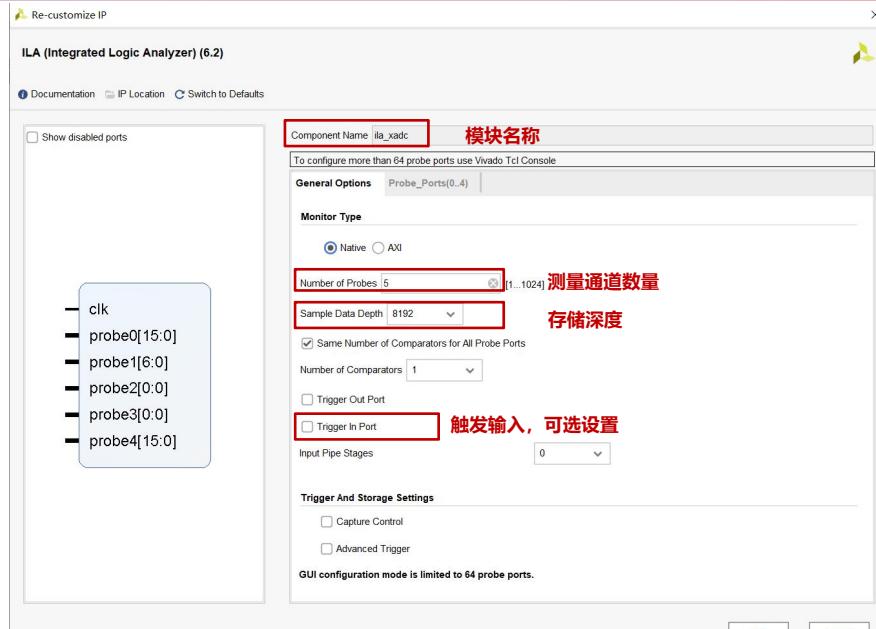
Vivado中ILA的使用



■ IP核方式使用方法

1. 创建IP核，计算并设置ILA相关参数
2. 在顶层文件中实例化ILA
3. 综合-实现-生成比特流文件
4. 烧录比特流文件到FPGA
5. 在waveform窗口中添加待观测信号
6. 软件触发获取FPGA信号

The screenshot shows the Vivado IP Catalog interface. A red arrow points to the 'IP Catalog' tab at the top. Below it, a search bar shows 'Search: ILA' with '(4 matches)' results. The results table lists several IP cores, with one entry for 'ILA (Integrated Logic Analyzer)' highlighted by a red arrow. Other entries include 'Multilayer Video Controller', 'Debug & Verification', 'System ILM', and 'Video & Image Processing'. The table columns include Name, AXI4, Status, License, and VLNIV.



Vivado中ILA的使用



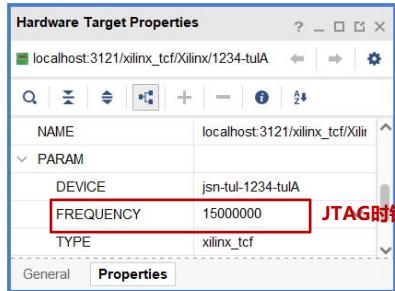
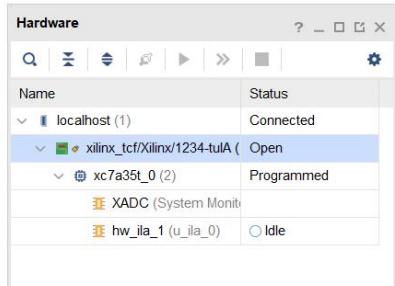
■ IP核方式使用注意事项

1. 选择合适的测量时钟:

频率太低无法抓取高速信号，频率太高存储时长变短

2. 测量时钟的频率必须大于JTAG时钟频率的2.5倍

3. 注意选择合适的触发信号：与使用示波器类似

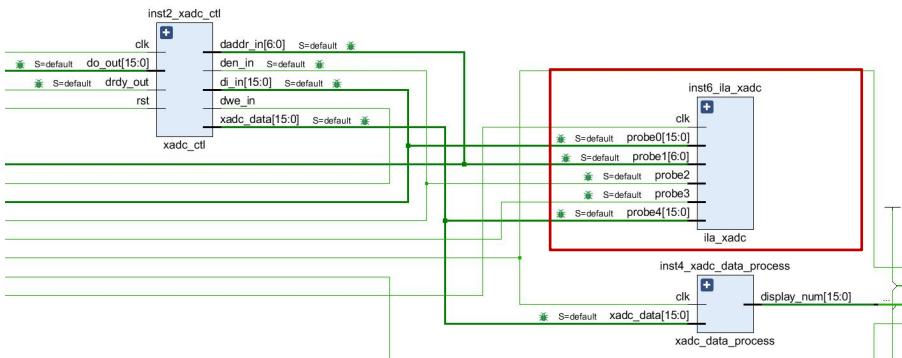


JTAG时钟频率 (Hz)

ila_xadc inst6_ilaxadc

(

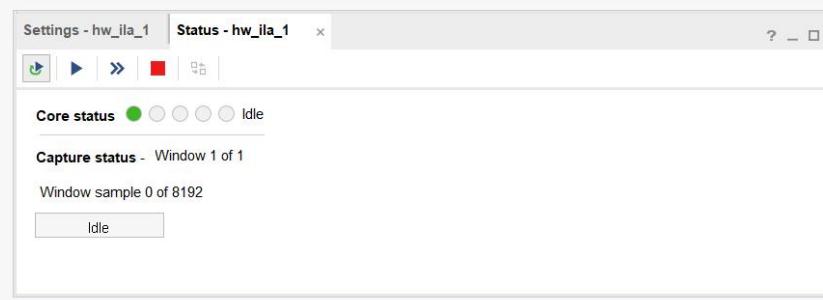
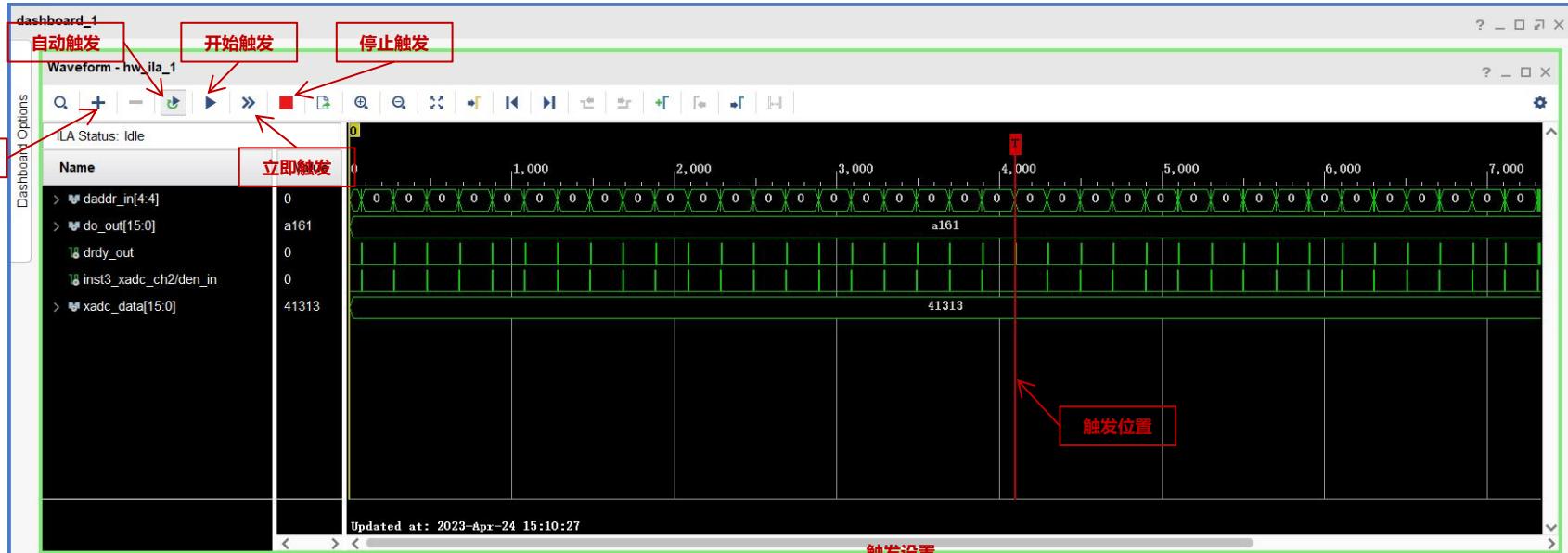
```
.clk(clk_50MHz), 模块时钟 // input wire clk
.probe0(di_in),           // input wire [15:0] probe0
.probe1(daddr_in),        // input wire [6:0]  probe1
.probe2(den_in),          // input wire [0:0]  probe2
.probe3(drdy_out),        // input wire [0:0]  probe3
.probe4(xadc_data)        // input wire [0:0]  probe4
);
```



Vivado中ILA的使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS



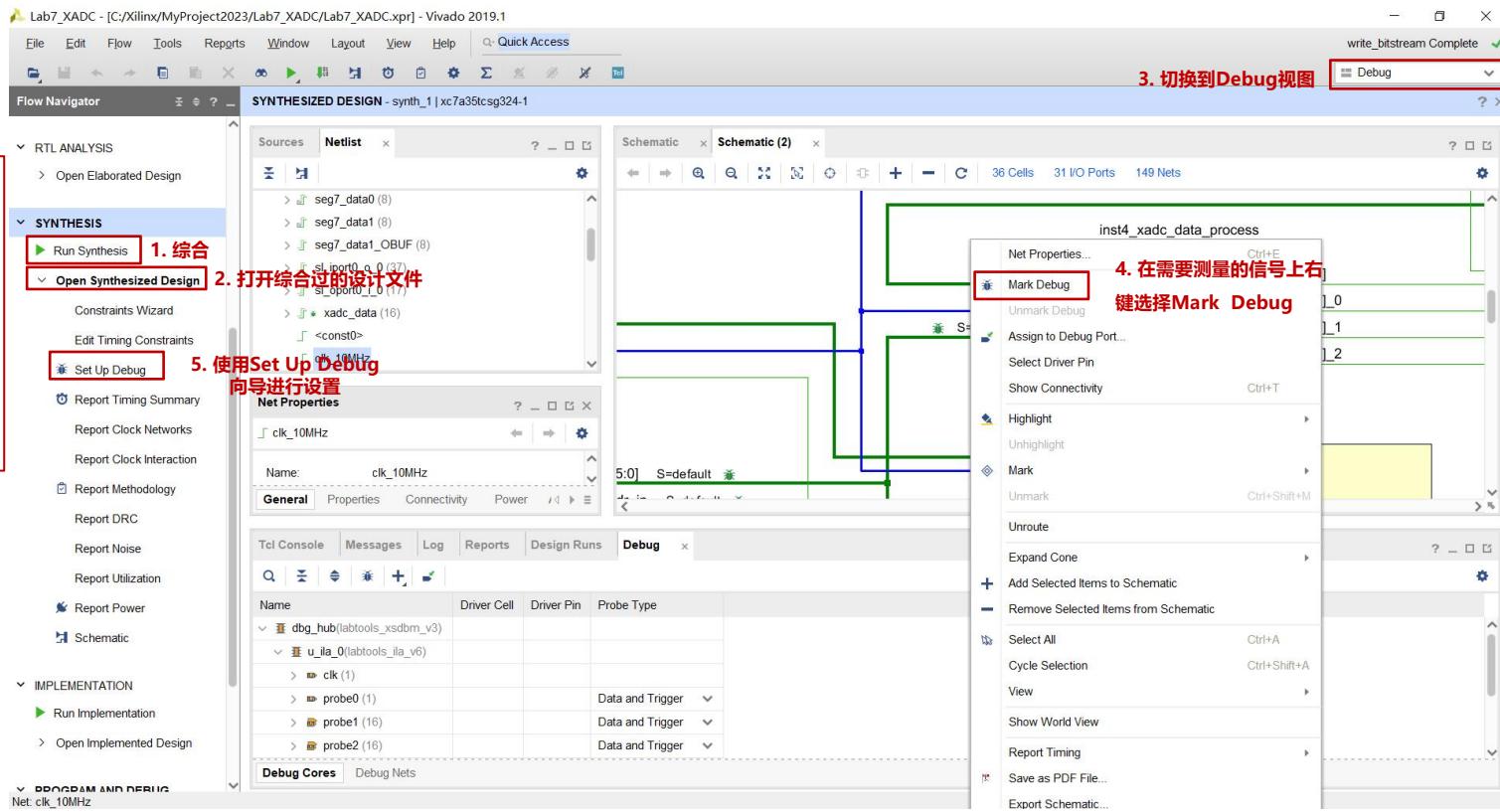
Vivado中ILA的使用



國科大杭州高華研究院 Hangzhou Institute for Advanced Study, UCAS

■ Set up Debug使用方法

1. 综合 (synthesis)
 2. 打开综合过的设计文件
 3. 切换到Debug视图
 4. 在需要测量的信号上右键选择
Mark Debug
 5. 通过Set Up Debug向导进行设置



Vivado中ILA的使用



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Set Up Debug

This wizard will guide you through the process of:

- Choosing nets and connecting them to debug cores.
- Associating a clock domain with each of the nets chosen for debug.
- Choosing additional features on the debug cores like Data Depth, Advanced Trigger mode and Capture Control.

Note: This setup wizard does not apply to the VIO, IBERT or JTAG-to-AXI-Master debug cores. Please refer to [Vivado Design Suite User Guide: Programming and Debugging \(UG908\)](#) for further instructions on how to use these IPs.

1. 向导说明

< Back Next > Finish Cancel

Set Up Debug

Nets to Debug

The nets below will be debugged with ILA cores. To add nets click "Find Nets to Add". You can also select nets in the Netlist or other windows, then drag them to the list or click "Add Selected Nets".

Name	Clock Domain	Driver Cell	Probe Type
> daddy_in (1)	inst1_pll/inst/clk_10MHz	FDCE	Data and Trigger
> do_out (16)	inst1_pll/inst/clk_10MHz	XADC	Data and Trigger
> xadc_data (16)	inst1_pll/inst/clk_10MHz	FDCE	Data and Trigger
> inst3_xadc_ch2/den_in	inst1_pll/inst/clk_10MHz	FDCE	Data and Trigger
> drdy_out	inst1_pll/inst/clk_10MHz	XADC	Data and Trigger

2. 信号选择，注意信号对应的时钟域

Find Nets to Add... Nets to debug: 35

< Back Next > Finish Cancel

Set Up Debug

ILA Core Options

Choose features for the ILA debug cores.

Sample of data depth: Input pipe stages:

3. 存储深度选择

Trigger and Storage Settings

Capture control
 Advanced trigger

Tcl Console Messages Log Reports Design Runs Debug

Debug Cores

Name	Driver Cell	Driver Pin	Probe Type
dbg_hub(labtools_xsdbm_v3)			
u ila_0(labtools_ila_v6)			
clk (1)			
probe0 (16)			Data and Trigger
probe1 (1)			Data and Trigger
probe2 (16)			Data and Trigger
probe3 (1)			Data and Trigger
probe4 (1)			Data and Trigger

Debug Nets

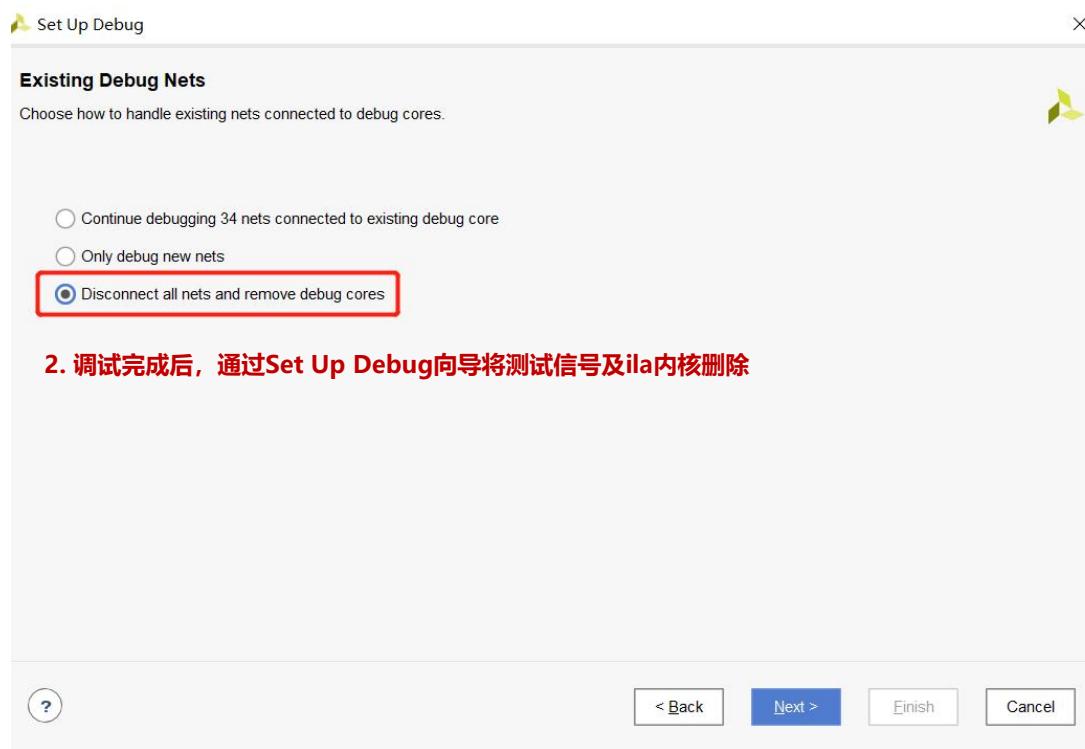
4. ila模块信息确认：时钟、通道

Vivado中ILA的使用

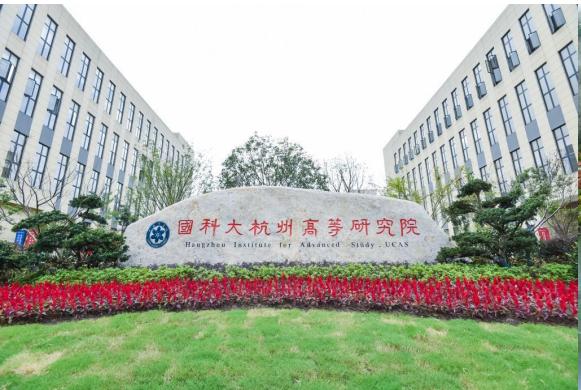


國科大杭州高茅研究院
Hangzhou Institute for Advanced Study,UCAS

1. 设置完成后Set Up Debug的后续使用方法与ILA IP核方式相同



2. 调试完成后，通过Set Up Debug向导将测试信号及ila内核删除



THANKS



國科大杭州高等研究院
Hangzhou Institute for Advanced Study, UCAS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第十讲 (10.2 XADC实验)

2025年5月6日

Lab8 XADC实验



■ 实验内容

- 通过XADC采集EGo1电路板上电位计的模拟电压
- 将采集到的模拟电压显示在数码管上
- 实验1：显示电位计的模拟电压
- 实验2：通过按键切换显示电位计模拟电压及片上温度

■ 实验目的

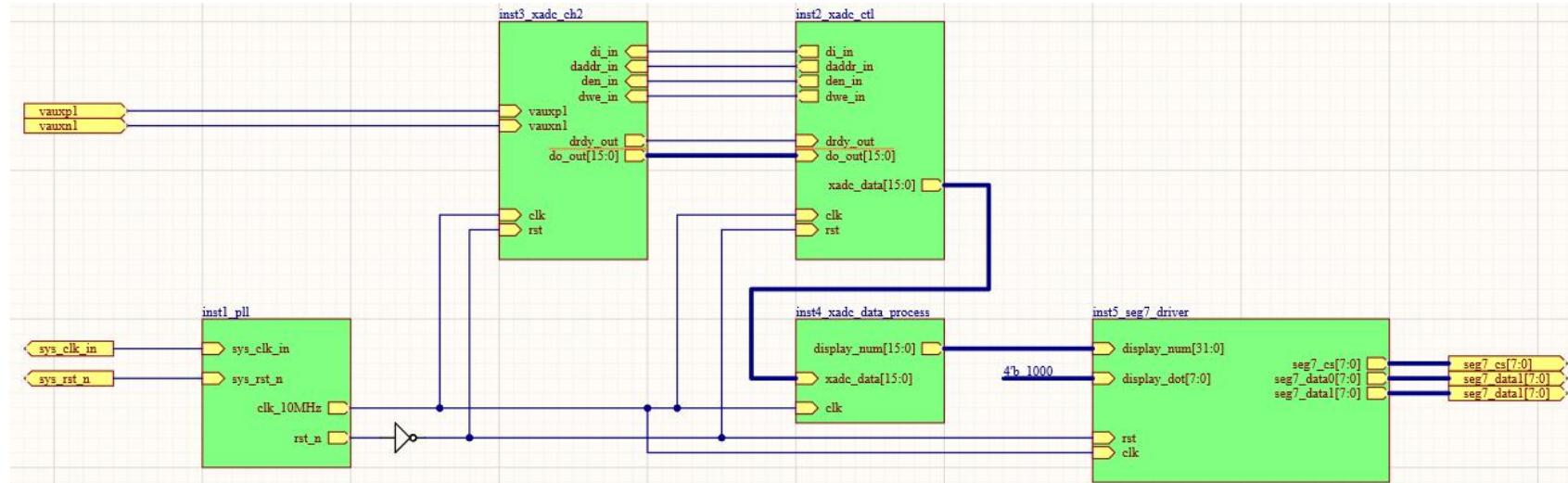
- 熟悉XADC的使用方法
- 学会ILA的使用
- 学会计算类（乘法及除法）IP核的使用



Lab8 XADC实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



■ 系统框图

- 电路共包括五个模块
- pll模块完成系统时钟管理，通过IP管理器配置、例化后实现
- xadc_ch2模块为2通道AD采集模块，通过IP管理器配置、例化后实现
- xadc_ctl模块为xadc_ch2时序控制模块，产生xadc工作所需的时序信号，采样得到xadc_data数据
- xadc_data_process模块为数据处理模块，完成二进制数据至BCD码转换
- seg7_driver模块为数码管驱动模块，完成xadc数据显示

Lab8 XADC实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

输入输出端口表

序号	信号名称	信号类型	输入/输出	信号描述	FPGA引脚	说明
1	sys_clk_in	1bit	输入	系统时钟输入	P17	100MHz输入
2	sys_rst_n	1bit	输入	系统复位输入	P15	低电平复位
3	switch_display	1bit	输入	数据切换输入	R1	高电平使能有效
4	switch_status	1bit	输出	数据状态显示	K2	高电平点亮
5	seg7_cs[7:0]	8bit	输出	数码管位选控制信号输出	G2/C2/C1/H1 G1/F1/E1/G6	高电平选通
6	seg7_data0[7:0]	8bit	输出	数码管字段控制信号输出	B4/A4/A3/B1 A1/B3/B2/D5	高电平有效
7	seg7_data1[7:0]	8bit	输出	数码管字段控制信号输出	D4/E3/D3/F4 F3/E2/D2/H2	高电平有效
8	vauxp1	1bit	输入	模拟电压输入	C12	电位计模拟信号输入
9	vauxn1	1bit	输入	模拟电压输入	B12	GND
10	vp_in	1bit	输入	模拟电压输入	J10	空载
11	vn_in	1bit	输入	模拟电压输入	K9	空载

Lab8 XADC实验



```
module Lab8_top(
    input sys_clk_in,
    input sys_rst_n,
    input switch_display,
    input vauxp1,
    input vauxn1,
    input vp_in,
    input vn_in,
    output [7:0] seg7_cs,
    output [7:0] seg7_data0,
    output [7:0] seg7_data1
);
```

```
wire clk_10MHz;
wire clk_50MHz;
wire rst;
wire [15:0] di_in;
wire [6:0] daddr_in;
wire den_in;
wire dwe_in;
wire drdy_out;
wire [15:0] do_out;
wire [15:0] xadc_data;
wire [15:0] display_num;
wire [7:0] display_dot;

assign rst = ~rst_n;
```

```
/////////////////////////////
```

```
pll inst1_pll
(
    .clk_10MHz(clk_10MHz),           // output clk_10MHz
    .clk_50MHz(clk_50MHz),           // input resetn
    .resetn(sys_rst_n),              // output rst_n
    .rst_n(rst_n),                  // input sys_clk_in
);
```

```
/////////////////////////////
```

```
xadc_ctl inst2_xadc_ctl
(
    .clk(clk_10MHz),
    .rst(rst),
    .di_in(di_in),
    .daddr_in(daddr_in),
    .den_in(den_in),
    .dwe_in(dwe_in),
    .switch_display(switch_display),
    .drdy_out(drdy_out),
    .do_out(do_out),
    .xadc_data(xadc_data)
);
```

```
///////////////////////////// xadc_ch2 inst3_xadc_ch2
(
    .di_in(di_in),                  // input wire [15 : 0] di_in
    .daddr_in(daddr_in),            // input wire [6 : 0] daddr_in
    .den_in(den_in),                // input wire den_in
    .dwe_in(dwe_in),                // input wire dwe_in
    .drdy_out(drdy_out),            // output wire drdy_out
    .do_out(do_out),                // output wire [15 : 0] do_out
    .clk_in(clk_10MHz),             // input wire clk
    .reset_in(rst),                 // input wire rst
    .vp_in(vp_in),                  // input wire vp_in
    .vn_in(vn_in),                  // input wire vn_in
    .vauxp1(vauxp1),                // input wire vauxp1
    .vauxn1(vauxn1),                // input wire vauxn1
    .vauxp9(),                      // input wire vauxp9
    .vauxn9(),                      // input wire vauxn9
    .channel_out(),                  // output wire [4 : 0] channel_out
    .eoc_out(),                      // output wire eoc_out
    .alarm_out(),                   // output wire alarm_out
    .eos_out(),                      // output wire eos_out
    .busy_out()                     // output wire busy_out
);
///////////////////////////// xadc_data_process inst4_xadc_data_process
(
    .clk(clk_10MHz),                // input clk_10MHz
    .xadc_data(xadc_data),           // input wire [15 : 0] xadc_data
    .display_num(display_num)        // output wire [15 : 0] display_num
);
///////////////////////////// seg7_driver inst5_seg7_driver
(
    .clk(clk_10MHz),                // input clk_10MHz
    .rst(rst),                      // input rst
    .display_num(display_num),        // input wire [31 : 0] display_num
    .display_dot(4'b10_00),           // input wire [7 : 0] display_dot
    .seg7_cs(seg7_cs),                // output wire [7 : 0] seg7_cs
    .seg7_data0(seg7_data0),          // output wire [7 : 0] seg7_data0
    .seg7_data1(seg7_data1)           // output wire [7 : 0] seg7_data1
);
endmodule
```

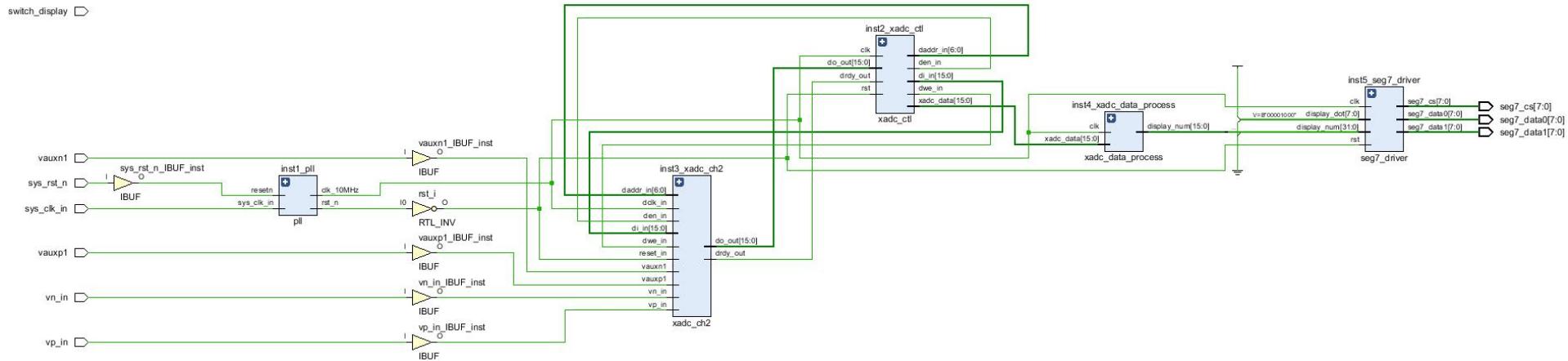
顶层文件(Lab8_top.v)
完成模块互联

Lab8 XADC实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

顶层文件(Lab8_top.v)
完成模块互联



Lab8 XADC实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

```
module xadc_ctl(
    input clk,
    input rst,
    output reg [15:0] di_in,
    output reg [6:0] daddr_in,
    output reg den_in,
    output reg dwe_in,
    input drdy_out,
    input [15:0] do_out,
    output reg [15:0] xadc_data
);

reg [7:0] counter;
wire marker_50KHz;
// 周期性采样信号, 50KHz, 10MHz/50KHz = 200
parameter COUNTER_50KHZ = 8'd200;

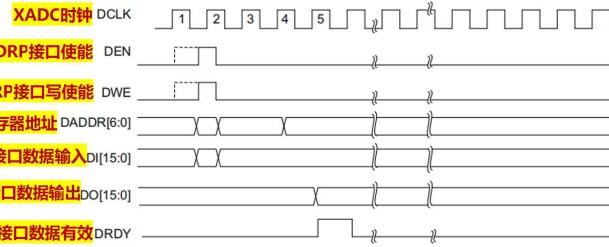
// 产生周期性采样信号
always @ (posedge clk or posedge rst) begin
    if(rst) begin
        counter <= 8'd0;
    end
    else if(counter < COUNTER_50KHZ) begin
        counter <= counter + 1'b1;
    end
    else begin
        counter <= 8'd0;      // 产生50KHz脉冲信号
    end
    assign marker_50KHz = (counter == 8'd200);
end

```

```
// 产生XADC时序
always @ (posedge clk or posedge rst) begin
    if(rst) begin
        di_in <= 16'd0;
        daddr_in <= 7'd0;
        den_in <= 1'b0;
        dwe_in <= 1'b0;
    end
    else if(marker_50KHz) begin
        di_in <= 16'd0;
        daddr_in <= 7'h11; // DRP接口送出地址11h
        den_in <= 1'b1;   // 对应为VAUX[1]数据地址
        dwe_in <= 1'b0;   // den信号置高一个时钟周期
    end
    else begin
        di_in <= 16'd0;
        daddr_in <= 7'd0;
        den_in <= 1'b0;
        dwe_in <= 1'b0;
    end
end

always @ (posedge clk or posedge rst) begin
    if(rst) begin
        xadc_data <= 16'd0;
    end
    else if(drdy_out) begin
        xadc_data <= do_out;
    end
    else begin
        xadc_data <= xadc_data;
    end
end
endmodule
```

等待DRP接口的DRDY信号为高
变高则采集一次数据
xadc_data为最新的XADC转换数据



XADC控制模块(xadc_ctl.v)
产生XADC所需的时序信号
采集XADC输出的数据

Status Registers (00h-3Fh)
Read Only

Temp (00h)	Temp Max (20h)
V_CCONT (01h)	V_CCONT Max (21h)
V_CCAUX (02h)	V_CCAUX Max (22h)
V_CCBRAH (03h)	V_CCBRAH Max (23h)
•	•
V_CCPIN1 (0Dh) (1)	Temp Min (24h)
V_CCPIN2 (0Eh) (1)	V_CCONT Min (25h)
V_CCBRAU (0Fh) (1)	V_CCAUX Min (26h)
V_CCO_DDR (0Fh) (1)	V_CCBRAH Min (27h)
VAUXP[0]/VAUXN[0] (10h)	V_CCPIN1 Max (28h) ⁽¹⁾
VAUXP[1]/VAUXN[1] (11h)	V_CCOIN Max (29h) ⁽¹⁾
•	•
VAUXP[12]/VAUXN[12] (1Ch)	VCCO_DDR Max (2Ah) ⁽¹⁾
VAUXP[13]/VAUXN[13] (1Dh)	Unassigned (2Bh)
VAUXP[14]/VAUXN[14] (1Eh)	V_CCPIN Min (2Ch) ⁽¹⁾
VAUXP[15]/VAUXN[15] (1Fh)	V_CCAUX Min (2Dh) ⁽¹⁾
Flag (3Fh)	V_CCBRAH Min (2Eh) ⁽¹⁾

Lab8 XADC实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Project Summary Schematic IP Cat

Cores | Interfaces

Search: Q- m

Name

- Dividers
 - Divider Generator
- Floating Point
 - Floating-point
- Multipliers
 - Complex Multiplier
 - Multiplier** 乘法IP核名称
- Square Root
- CORDIC
- Trig Functions
- CORDIC

Component Name: mult_gen

Basic Output and Control

Multiplier Type

Parallel Multiplier Constant Coefficient Multiplier

Input Options

P = A * B

Data Type: Unsigned Unsigned

Width: 16 16

参数:
(1) 数据类型: 有符号, 无符号
(2) 数据位宽

Multiplier Construction: Use Mults

Optimization Options: Speed Optimized

Area: Optimizes the multiplier for DSP48 slice resources by splitting the multiplication between DSP48 slices and Speed: Optimizes the multiplier for performance using as many DSP48 slices as necessary

乘法IP核

Multiplier v12.0

```
mult_gen inst1
(
    IP核时钟 .CLK(clk),
    乘数A .A(xadc_data),
    乘数B .B(16'd1000),
    结果P .P(xadc_data_x1000)
);
```

LogiCORE IP Product Guide

Vivado Design Suite

PG108 November 18, 2015

Lab8 XADC实验



除法IP核

Project Summary | Schematic | IP Catalog

Cores | Interfaces



Search:

Name

Vivado Repository

Math Functions

Dividers

Divider Generator

除法IP核名称

```
div_gen inst2_div_gen
(
    .aclk(clk), IP核时钟
    .s_axis_divisor_tvalid(1'b1), 除数有效信号
    .s_axis_divisor_tdata(16'd1000), 除数
    .s_axis_dividend_tvalid(1'b1), 被除数有效信号
    .s_axis_dividend_tdata(xadc_data_x1000[31:16]), 被除数
    .m_axis_dout_tvalid(),
    .m_axis_dout_tdata({thousand_quotient, thousand_remainder}) 商+余数
);
```

Component Name: div_gen

Channel Settings | Options

Common Options

Algorithm Type: Radix2

Operand Sign: Unsigned

Dividend Channel

Dividend Width: 16 [2 - 64] **被除数位宽**

Has TLAST Has TUSER

TUSER Width: 1 [1 - 256]

Divisor Channel

Divisor Width: 16 [2 - 64] **除数位宽**

Has TLAST Has TUSER

TUSER Width: 1 [1 - 256]

Output Channel

Remainder Type: Remainder **余数类型**

Fractional Width: 16 [0 - 64]

Detect Divide-By-Zero Number of Iterations = n/a Throughput = n/a

Divider Generator v5.1

LogiCORE IP Product Guide

Vivado Design Suite

PG151 February 4, 2021

Lab8 XADC实验



```
module xadc_data_process(
    input clk,
    input [15:0] xadc_data,
    output [15:0] display_num
);

// 将xadc_data转换为mv值, xdata_data * 1000/4096
wire [31:0] xadc_data_x1000;
wire [15:0] thousand_quotient,hundred_quotient,ten_quotient,thousand_remainder,hundred_remainder,ten_remainder;

mult_gen inst1
(
    .CLK(clk),
    .A(xadc_data), // (1) 得到xadc_data_x1000
    .B(16'd1000),
    .P(xadc_data_x1000)
);

div_gen inst2_div_gen (3) 得到千位
(
    .aclk(clk),
    .s_axis_divisor_tvalid(1'b1),
    .s_axis_divisor_tdata(16'd1000),
    .s_axis_dividend_tvalid(1'b1),
    .s_axis_dividend_tdata(xadc_data_x1000[31:16]), // (2) 数据移位16bit
    .m_axis_dout_tvalid(),
    .m_axis_dout_tdata((thousand_quotient,thousand_remainder))
);

div_gen inst3_div_gen (4) 得到百位
(
    .aclk(clk),
    .s_axis_divisor_tvalid(1'b1),
    .s_axis_divisor_tdata(16'd100),
    .s_axis_dividend_tvalid(1'b1),
    .s_axis_dividend_tdata(thousand_remainder),
    .m_axis_dout_tvalid(),
    .m_axis_dout_tdata((hundred_quotient,hundred_remainder))
);

div_gen inst4_div_gen (5) 得到十位及个位
(
    .aclk(clk),
    .s_axis_divisor_tvalid(1'b1),
    .s_axis_divisor_tdata(16'd10),
    .s_axis_dividend_tvalid(1'b1),
    .s_axis_dividend_tdata(hundred_remainder),
    .m_axis_dout_tvalid(),
    .m_axis_dout_tdata((ten_quotient,ten_remainder))
);

assign display_num = {thousand_quotient[3:0],hundred_quotient[3:0],ten_quotient[3:0],ten_remainder[3:0]}; // (6) 数据合并
endmodule
```

XADC数据处理模块
(xadc_data_process.v)
将XADC数据转换为BCD码数据

$$Voltage(mV) = \frac{ADC\ code \times 1000}{4096}$$

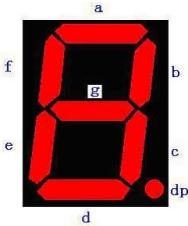
DI15	DI14	DI13	DI12	DI11	DI10	DI9	DI8	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0	DATA[11:0]	Note

■ 数据处理流程

- (1) xadc_data乘1000, 得到xadc_data_x1000
- (2) xadc_data_x1000除4096 (相当于12位右移位) , 因12bit数据在xadc_data的高12位上, 需要做4位右移位, 因此合计右移位16bit, 得到mv为单位的二进制码值
- (3) 将该二进制码值除1000, 商为BCD的千位, 得到余数
- (4) 将上步的余数除以100, 商为BCD码的百位, 得到余数
- (5) 将上步的余数除以10, 商为BCD码的十位, 余数为BCD码的个位
- (6) 取千位、百位、十位、各位的低4位合并为display_num

分别使用了乘法及除法IP核实现

Lab8 XADC实验



根据数码管字库的定义，小数点为字段信号的最高位，需要显示小数点时相当于在数字字段上或
0x80(|8'h80)

```
module seg7_driver(
    input clk,
    input rst,
    input [31:0] display_num,
    input [7:0] display_dot,
    output reg [7:0] seg7_cs,
    output [7:0] seg7_data0,
    output [7:0] seg7_data1
);

// 定义小数点显示标记
reg dot_marker;
```

```
case(led_num)
  4'd0: begin
    current_display_num <= display_num[3:0];
    seg7_cs = CS0;
    if(display_dot[0] == 1'b1) begin
      dot_marker <= 1'b1;
    end
    else begin
      dot_marker <= 1'b0;
    end
  end
  4'd1: begin
    current_display_num <= display_num[7:4];
    seg7_cs <= CS1;
    if(display_dot[1] == 1'b1) begin
      dot_marker <= 1'b1;
    end
    else begin
      dot_marker <= 1'b0;
    end
  end
  4'd2: begin
    current_display_num <= display_num[11:8];
    seg7_cs <= CS2;
    if(display_dot[2] == 1'b1) begin
      dot_marker <= 1'b1;
    end
    else begin
      dot_marker <= 1'b0;
    end
  end
end
```

```
//根据数码管显示数据进行字库索引
always @ (posedge clk or posedge rst) begin
  if(rst) begin
    data_out <= NUM0;
  end
  else begin
    case(current_display_num)
      4'h0: data_out <= NUM0;
      4'h1: data_out <= NUM1;
      4'h2: data_out <= NUM2;
      4'h3: data_out <= NUM3;
      4'h4: data_out <= NUM4;
      4'h5: data_out <= NUM5;
      4'h6: data_out <= NUM6;
      4'h7: data_out <= NUM7;
      4'h8: data_out <= NUM8;
      4'h9: data_out <= NUM9;
      4'ha: data_out <= NUMA;
      4'hb: data_out <= NUMB;
      4'hc: data_out <= NUMC;
      4'hd: data_out <= NUMD;
      4'he: data_out <= NUME;
      4'hf: data_out <= NUMF;
      default: data_out <= NUM0;
    endcase
  end
end
|
dot_marker有效，当前显示的数据上加小数点
assign seg7_data0 = (dot_marker) ? (data_out | 8'h80) : data_out;
```

数码管显示小数点的一种方法

- (1) 通过display_dot控制每一位数码管小数点的状态
- (2) 动态刷新显示时根据display_dot产生dot_marker信号
- (3) 根据dot_marker确定当前显示数据是否要加小数点



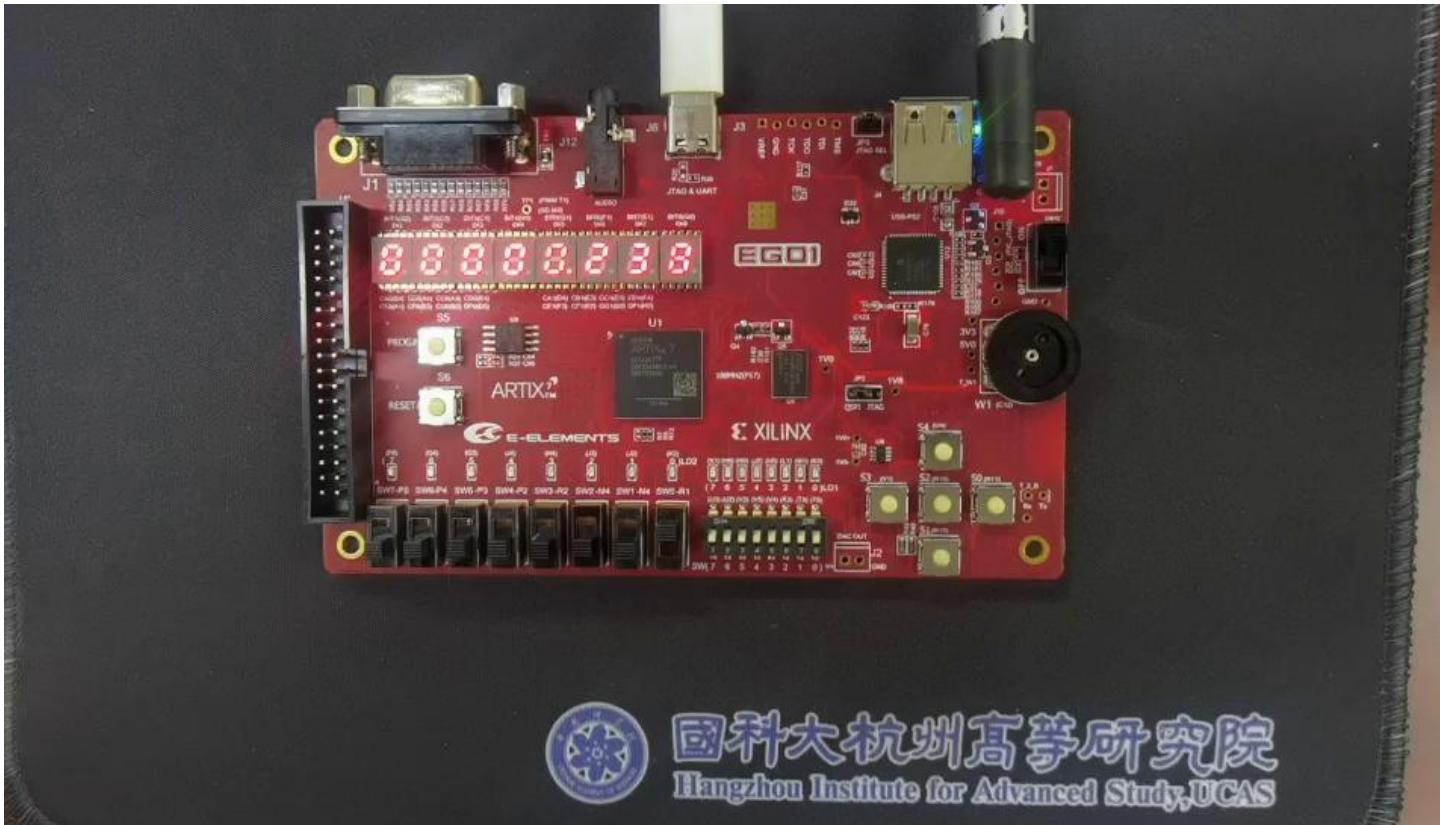
附加实验

实现FPGA片上温度传感器的信号采集

Lab8 XADC实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Lab8 XADC实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

■ 顶层修改

1. xadc_ctl模块:

接入switch_display控制信号

根据控制信号读取xadc对应地址的寄存器数据

2. xadc_data_process模块:

接入switch_display控制信号

根据控制信号对xadc输出的数据进行处理

输出display_dot信号

根据switch_display信号输出对应的display_dot

```
////////////////////////////////////////////////////////////////  
xadc_ctl inst2_xadc_ctl  
(  
    .clk(clk_10MHz),  
    .rst(rst),  
    .di_in(di_in),  
    .daddr_in(daddr_in),  
    .den_in(den_in),  
    .dwe_in(dwe_in),  
    .switch_display(switch_display),  
    .drdy_out(drdy_out),  
    .do_out(do_out),  
    .xadc_data(xadc_data)  
)  
xadc_data_process inst4_xadc_data_process  
(  
    .clk(clk_10MHz),  
    .xadc_data(xadc_data),  
    .switch_display(switch_display),  
    .display_dot(display_dot),  
    .display_num(display_num)  
)  
////////////////////////////////////////////////////////////////  
seg7_driver inst5_seg7_driver  
(  
    .clk(clk_10MHz),  
    .rst(rst),  
    .display_num(display_num),  
    .display_dot(display_dot),  
    .seg7_cs(seg7_cs),  
    .seg7_data0(seg7_data0),  
    .seg7_data1(seg7_data1)  
)
```

Lab8 XADC实验



■ xadc_ctl模块

1. 接入switch_display控制信号
2. switch_display==1时，读取VAUXP[1]的数据
switch_display==0时，读取片内温度的数据

Status Registers (00h–3Fh)
Read Only

Temp (00h)	Temp Max (20h)
V _{CCINT} (01h)	V _{CCINT} Max (21h)
V _{CCAOX} (02h)	V _{CCAOX} Max (22h)
V _{p/V_N} (03h)	V _{CCBRAM} Max (23h)
•	Temp Min (24h)
V _{CCPINT} (0Dh) (1)	V _{CCINT} Min (25h)
V _{CCPAUX} (0Eh) (1)	V _{CCAOX} Min (26h)
V _{CCO_DDR} (0Fh) (1)	V _{CCBRAM} Min (27h)
VAUXP[0]/VAUXN[0] (10h)	V _{CCPINT} Max (28h) ⁽¹⁾
VAUXP[1]/VAUXN[1] (11h)	V _{CCPAUX} Max (29h) ⁽¹⁾
•	V _{CCO_DDR} Max (2Ah) ⁽¹⁾
VAUXP[12]/VAUXN[12] (1Ch)	Unassigned (2Bh)
VAUXP[13]/VAUXN[13] (1Dh)	V _{CCPINT} Min (2Ch) ⁽¹⁾
VAUXP[14]/VAUXN[14] (1Eh)	V _{CCPAUX} Min (2Dh) ⁽¹⁾
VAUXP[15]/VAUXN[15] (1Fh)	V _{CCO_DDR} Min (2Eh) ⁽¹⁾
	Unassigned (2Fh)
	Undefined (30h)
	Undefined (31h)
	Flag (3Fh)

```
// 产生XADC时序
always @(posedge clk or posedge rst) begin
    if(rst) begin
        di_in <= 16'd0;
        daddr_in <= 7'd0;
        den_in <= 1'b0;
        dwe_in <= 1'b0;
    end
    else if(marker_50KHz) begin
        di_in <= 16'd0;
        if(switch_display) begin
            daddr_in <= 7'h11;
        end
        else begin
            daddr_in <= 7'h00;
        end
        den_in <= 1'b1;
        dwe_in <= 1'b0;
    end
    else begin
        di_in <= 16'd0;
        daddr_in <= 7'd0;
        den_in <= 1'b0;
        dwe_in <= 1'b0;
    end
end
```

Lab8 XADC实验



■ xadc_data_process模块

1. 增加温度转换计算
2. 复用二进制转BCD代码
3. 增加display_dot输出

```
// 将xadc_data转换为mv值, Voltage(mv) = xdata_data * 1000/4096
wire [31:0] xadc_data_x1000;
wire [31:0] xadc_data_x50398;
wire [15:0] xadc_data_temp;
wire [15:0] xadc_data_bin;
wire [15:0] thousand_quotient,hundred_quotient,ten_quotient,thousand_remainder,hundred_remainder,ten_remainder;

assign xadc_data_temp = xadc_data_x50398[31:16] - 16'd27315; 2. 除4096 (移位16位), 减27315
assign xadc_data_bin = switch_display? xadc_data_x1000[31:16] : xadc_data_temp;
3. 复用二进制转BCD代码

mult_gen inst1_mult_gen
(
    .CLK(clk),
    .A(xadc_data),
    .B(16'd1000),
    .P(xadc_data_x1000)
); // input wire CLK // input wire [15 : 0] A // input wire [15 : 0] B // output wire [31 : 0] P

mult_gen inst2_mult_gen
(
    .CLK(clk),
    .A(xadc_data),
    .B(16'd50398),
    .P(xadc_data_x50398)
); // input wire CLK // input wire [15 : 0] A // input wire [15 : 0] B // output wire [31 : 0] P

1. 将计算公式整体扩大100倍 (x100)
计算乘50398后的值

div_gen inst3_div_gen
(
    .aclk(clk),
    .s_axis_divisor_tvalid(1'b1),
    .s_axis_divisor_tdata(16'd1000),
    .s_axis_dividend_tvalid(1'b1),
    .s_axis_dividend_tdata(xadc_data_bin),
    .m_axis_dout_tvalid(),
    .m_axis_dout_tdata({thousand_quotient,hundred_quotient,ten_quotient,thousand_remainder,hundred_remainder,ten_remainder})
); // input wire aclk // input wire s_axis_divisor_tvalid // input wire [15 : 0] s_axis_divisor_tdata 除数 // input wire s_axis_dividend_tvalid // input wire [15 : 0] s_axis_dividend_tdata 被除数 // output wire m_axis_dout_tvalid // output wire [31 : 0] m_axis_dout_tdata [31:16]-商, [15:0]-余数

assign display_num = {thousand_quotient[3:0],hundred_quotient[3:0],ten_quotient[3:0],ten_remainder[3:0]};
assign display_dot = switch_display ? 7'b000_1000 : 7'b000_0100; 4. 正确放置温度小数点, 相当于完成了除100
```

下节预告



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

Lab9实验内容

Lab9串口发送实验

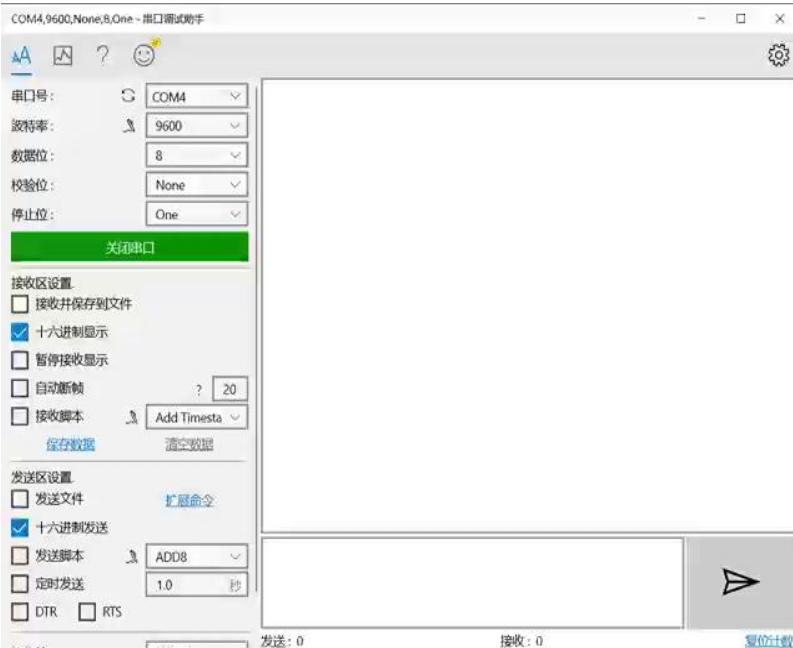


■ 实验内容

- 使用FPGA串口输出学号并通过上位机接收显示
- 学号数据产生模块循环产生学号数据，存入到FIFO中
- 串口模块根据FIFO的状态（非空），读取数据后串行发送
- 上位机接收串口循环输出的学号数据

■ 实验目的

- 学会使用状态机，并将其应用于数据发生器、串口控制器
- 熟悉FIFO及IP核的使用
- 熟悉UART的时序及使用

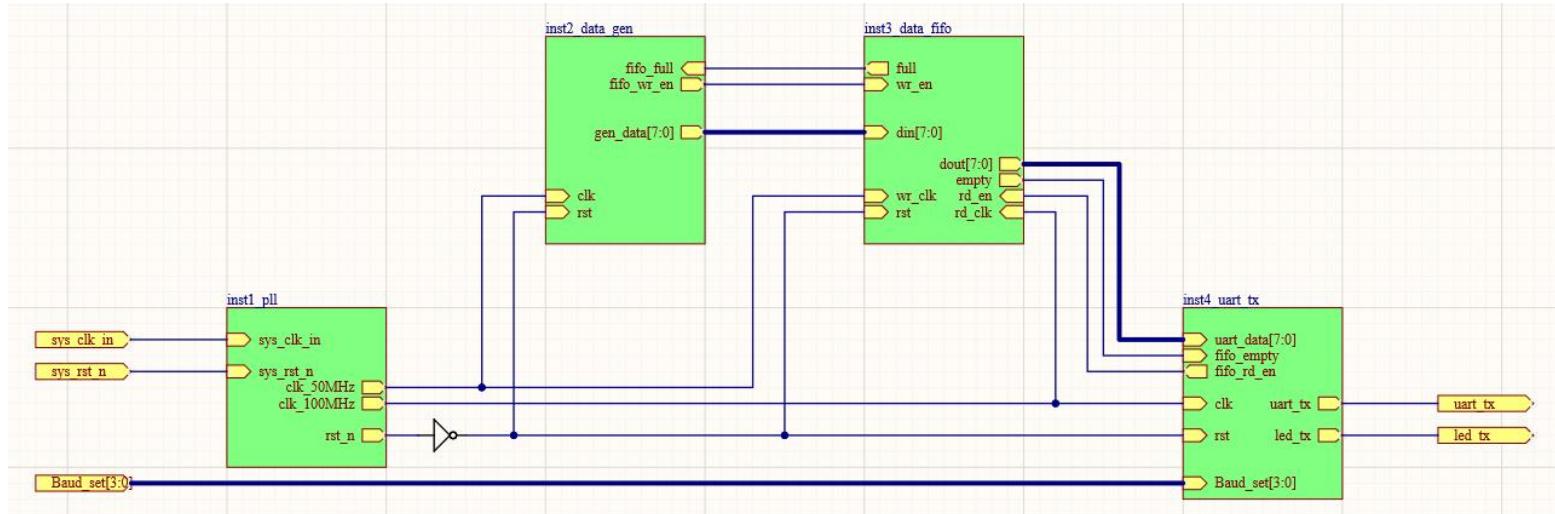


串口调试助手：可以在Micrsoft Store中搜索串口调试助手下载后使用

Lab9串口发送实验



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



■ 电路结构

- 系统包括四个模块，时钟管理模块、数据产生模块、FIFO模块、串口发送模块
- 时钟模块产生50MHz及100MHz时钟，通过FIFO实现跨时钟域数据传输
- 数据产生模块中使用状态机产生学号数据序列，根据FIFO的满标志状态，将产生的数据写入FIFO
- FIFO模块使用Vivado的IP核产生，设置为异步FIFO，数据8位，数据深度为512
- 串口发送模块产生波特率时钟，使用状态机判断FIFO空标志，将从FIFO读出的并行数据转化为串行输出



THANKS



國科大杭州高等研究院
Hangzhou Institute for Advanced Study, UCAS



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



FPGA电路软硬件设计 第十一讲 (11.1 FPGA设计技巧)

2025年5月13日



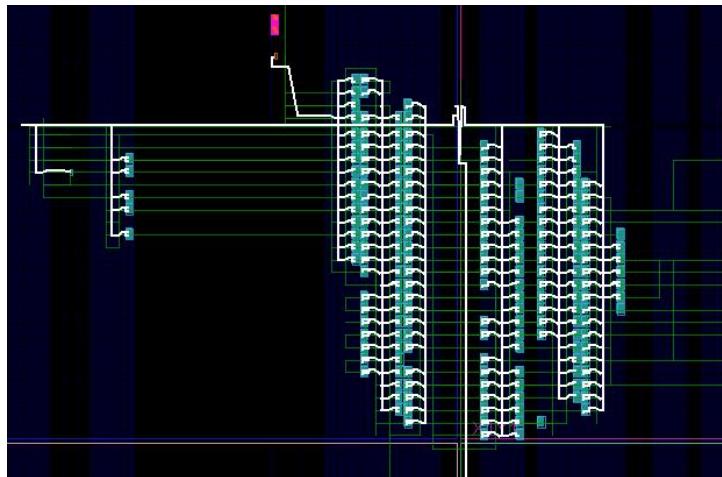
1. 时钟设计
2. 复位设计
3. 速度与面积的平衡与互换
4. 流水线
5. 层次化设计
6. 参数化设计
7. 调试技巧
8. Vitis软件平台
9. FPGA高级应用

1.时钟设计



■ 时钟设计原则

- 避免异步设计，**时钟是同步设计的脉搏**
- **时钟域：**时钟作用的范围，相当于时钟的“势力范围”
- **时钟树：**FPGA内部的时钟网络资源，如：**全局时钟、区域时钟、IO时钟**
- **原则一：**尽可能采用单一时钟
- **原则二：**尽可能利用FPGA内部的MMCM、PLL来作为时钟源
- **原则三：**避免使用“门生时钟/行波时钟”
- **原则四：**慎用“门控时钟”
- **原则五：**做好跨时钟域信号的处理

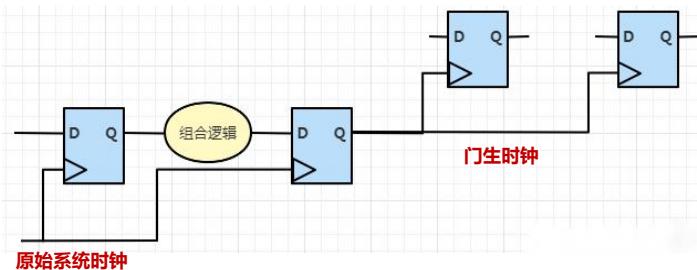


1.时钟设计



■ 避免使用内部逻辑产生的时钟（门生时钟/行波时钟）

- 容易导致功能**尤其是时序**出现较大问题
- 会引入新的时钟域
- 由于T_{co}、T_{data}、走线延迟，门生时钟会滞后于原始系统时钟
- 增加时序分析的难度，降低系统的可靠性
- 建议：①采用MMCM/PLL生成所需的时钟信号
②采用与主时钟同步的控制信号（采用课程实验中类似的mark信号）

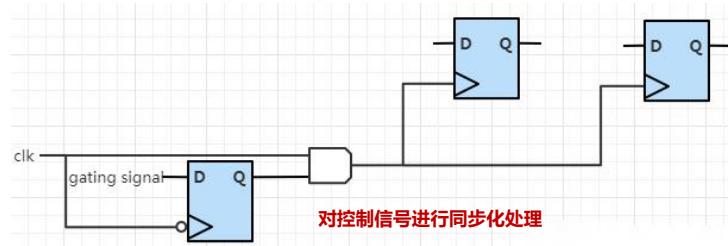
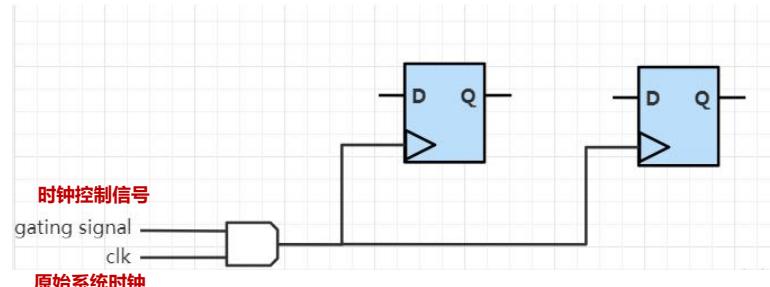


1.时钟设计



■慎用门控时钟

- 常用于芯片设计，通过**关闭时钟**，降低后级电路的**动态功耗**
- 有可能产生竞争与冒险，导致后续电路不稳定
- 解决方法：对控制信号进行**同步化处理**，由于**T_{co}**的存在，仍然有可能**产生毛刺**

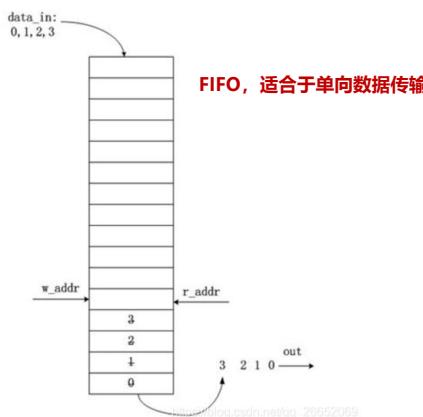


1.时钟设计

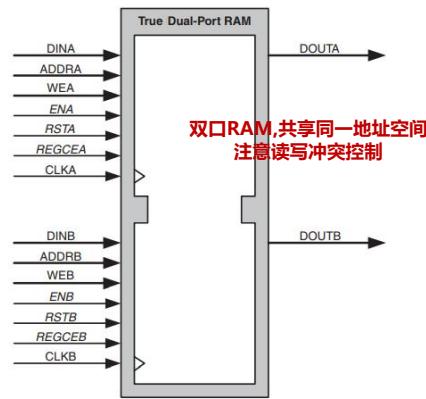


■ 跨时钟域信号的处理

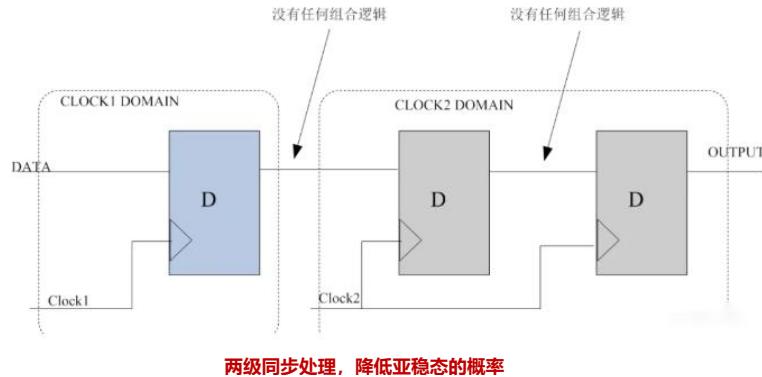
- 跨时钟域的时钟分类：慢速时钟->快速时钟，快速时钟->慢速时钟
- 跨时钟域的信号分类：单bit数据，多bit数据
- 单bit数据，慢速时钟->快速时钟：采用两级同步处理、握手机制
- 单bit数据，快速时钟->慢速时钟：采用握手机制
- 多bit数据，慢速时钟->快速时钟：FIFO、双口RAM、握手机制
- 多bit数据，快速时钟->慢速时钟：FIFO、双口RAM、握手机制



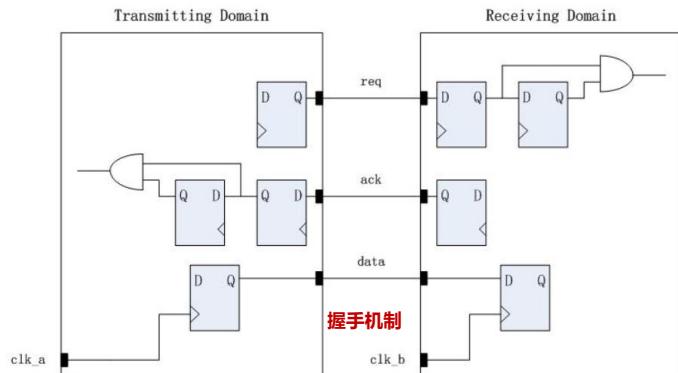
FIFO, 适合于单向数据传输



双口RAM, 共享同一地址空间
注意读写冲突控制

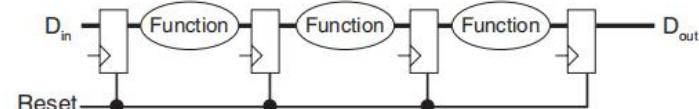


两级同步处理, 降低亚稳定的概率



1. 发送域将data有效;
2. 发送域将req信号有效, 通知接收域读取信号;
3. 接收域读取数据后, 接收域发送ack信号, 通知发送域读取结束

2. 复位设计



流水线，没有反馈，可以不采用复位设计
Figure 4: Reset for a Pipeline

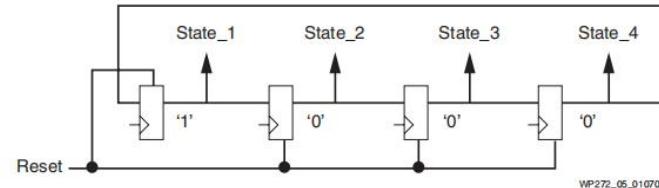
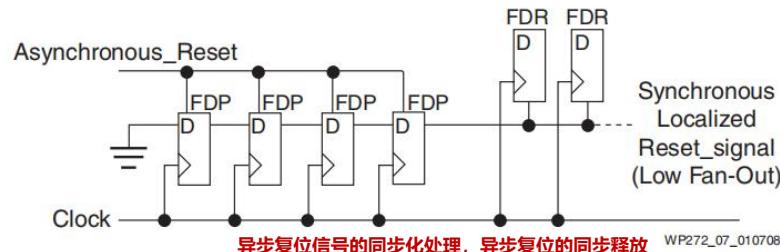


Figure 5: Reset for a One-Hot State Machine
状态机，有反馈，需要采用复位设计



异步复位信号的同步化处理，异步复位的同步释放
Figure 7: Localized Reset

3.速度与面积的平衡与互换



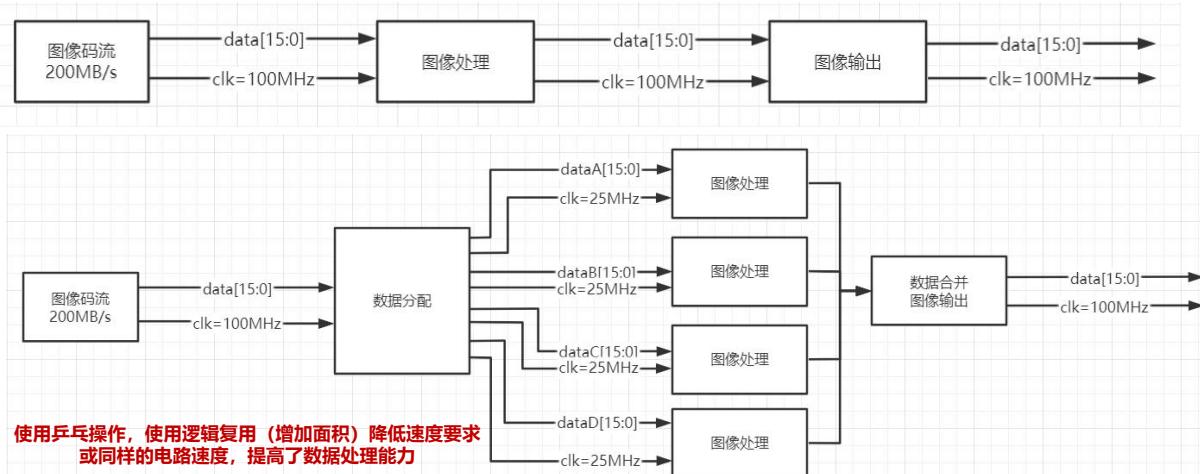
■速度与面积的平衡与互换

- 面积：消耗FPGA的逻辑资源（FF与LUT）的数量
- 速度：FPGA稳定运行所能达到的最高频率
- 设计目标：满足时序要求，占用面积最小；或满足面积要求，运行频率更快
- 互换：时序裕量大，可以通过模块复用减小消耗的面积；时序裕量小，需要通过流水线、乒乓操作等增加面积换取速度



■乒乓操作

- 用于数据流处理
- 通过数据分配单元，将数据等时分配到数据缓冲单元，进行同样方式的处理
- 实现低速模块处理高速数据或提高模块数据处理能力

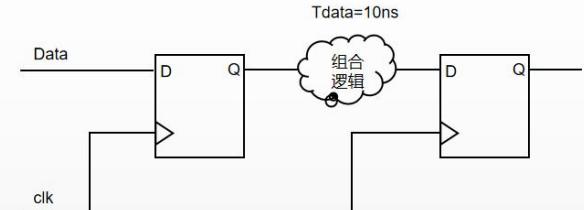


4.流水线



■ 流水线思想

- 拆解寄存器间的组合逻辑，中间加入更多级的寄存器
- 降低每级间的Tdata，显著提高电路性能
- 注意流水线带来的数据延迟



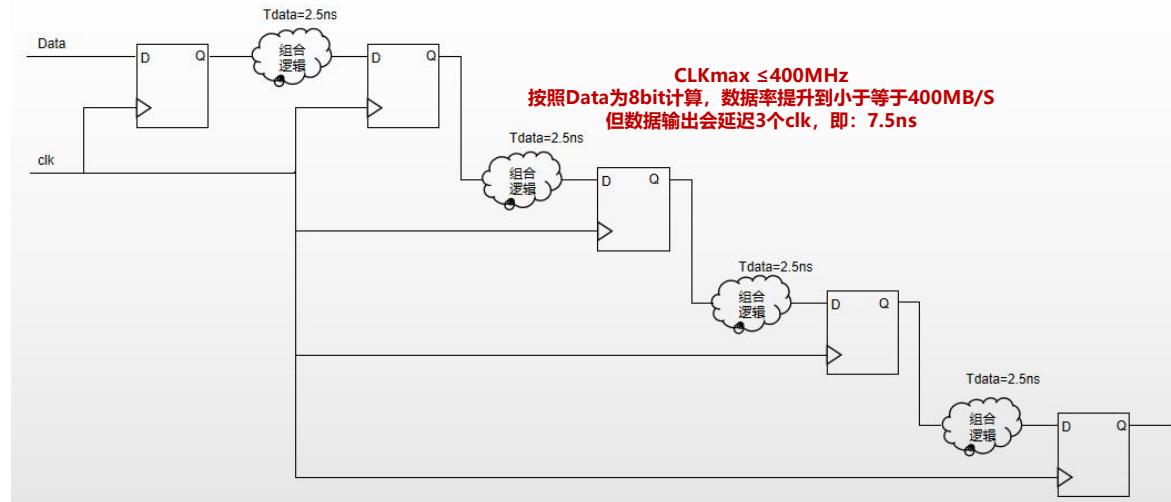
CLKmax ≤ 100MHz
按照Data为8bit计算，数据率小于等于100MB/S

Setup slack = Tclk - Tsu - Tco - Tdata

Tco,Tsu为器件固有特性

减小Tdata可以增大建立时间裕量

即电路主频可以跑得更快

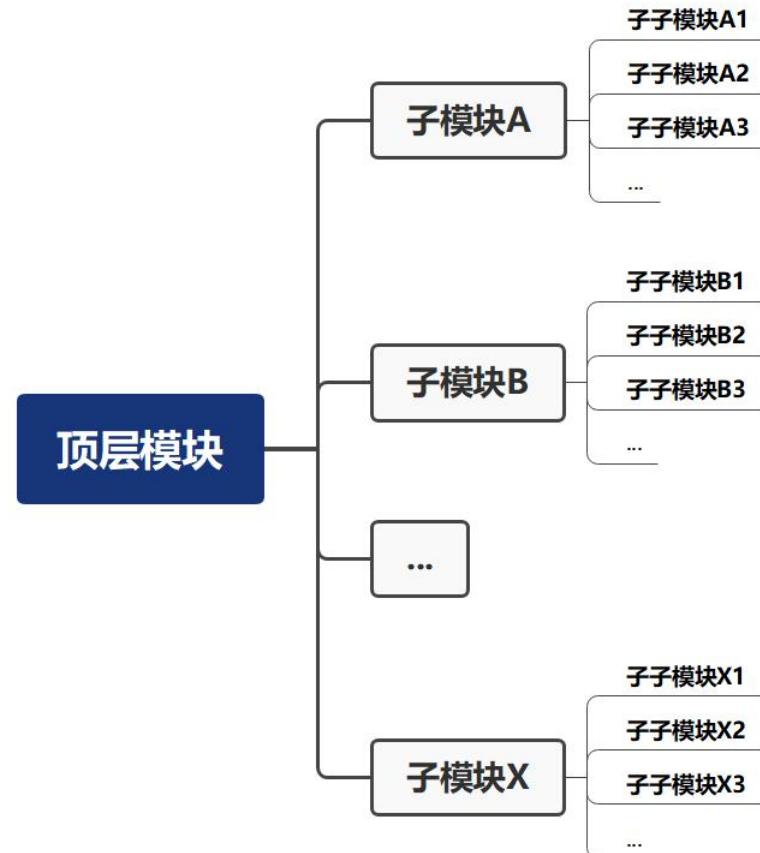


5. 层次化设计



■ 层次化设计

- 自顶而下，逐层划分子模块
- 是构建复杂电路系统的需要，团队协作，模块复用
- 便于代码仿真，降低调试难度
- 便于后期维护，代码优化，功能升级



6.参数化设计



■参数化设计

- 模块内部采用参数化设计
- 顶层对模块采用参数化例化
- 参数放入统一的define文件
- 优点：便于程序的维护与复用

```
module data_gen
#(
    parameter COUNTER_100MS = 32'd5_000_000,
    parameter BPS_CNT_MAX = 16'd10471
)
(
    input clk,
    input rst,
    input full,
    output reg wr_en,
    output reg [7:0] gen_data
);
```

模块内部采用参数化设计
该语法可以实现模块参数化例化

```
`include "parameter_define.v"
```

全局参数可以放入define文件

```
data_gen #(
    .COUNTER_100MS(`COUNTER_100MS),
    .BPS_CNT_MAX(`BPS_CNT_MAX)
)
inst2_data_gen
(
    .clk(clk_50MHz),
    .rst(rst),
    .full(fifo_full),
    .wr_en(fifo_wr_en),
    .gen_data(gen_data)
);
```

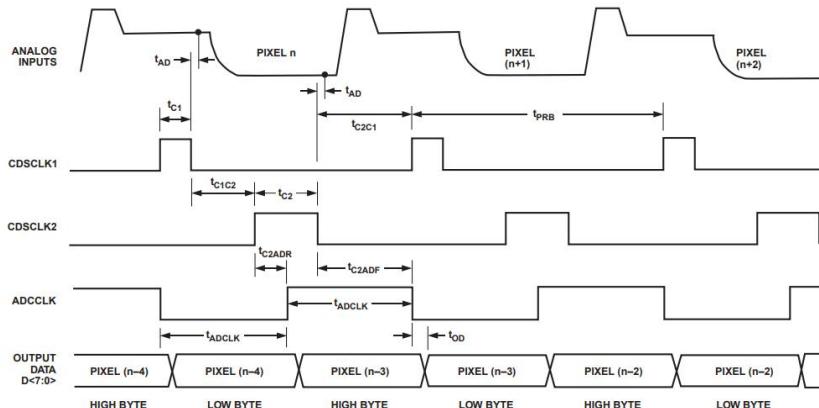
对模块进行参数化例化

```
`define COUNTER_100MS 32'd5_000_000
`define BPS_CNT_MAX 16'd5208
```

7. 调试技巧

■ 调试技巧

- 学会看datasheet，尤其是要习惯看英文原版
- 学会站在巨人的肩膀上（IP核，开源代码）
- 代码设计完成后，首先要查看RTL原理图
- 复杂代码在设计时就要想好测试用例，以及调试用信号
- 自己是最好的仿真测试工具，学会代码走查
- 熟练使用软件工具（vivado软件、嵌入式逻辑分析仪等）
- 学会使用板级调试工具（信号发生器、示波器、逻辑分析仪、万用表等）测量关键信号
- 重视写文档，不要厌烦写文档



TIMING SPECIFICATIONS (t_{MIN} to t_{MAX} , AVDD = 5 V, DVDD = 5 V, specs are for 16-bit performance.)

Parameter	Symbol	Min	Typ	Max	Unit
CLOCK PARAMETERS					
3-Channel Pixel Rate	t_{PRA}	200			ns
1-Channel Pixel Rate	t_{PRB}	80			ns
ADCCLK Pulsewidth	t_{ADCLK}	30			ns
CDSCLK1 Pulsewidth	t_{C1}	8			ns
CDSCLK2 Pulsewidth	t_{C2}	8			ns
CDSCLK1 Falling to CDSCLK2 Rising	t_{C1C2}	0			ns
ADCCLK Falling to CDSCLK2 Rising	t_{ADC2}	0	注意细节，时序要求等		
CDSCLK2 Rising to ADCCLK Rising	t_{C2ADR}	5			ns
CDSCLK2 Falling to ADCCLK Falling	t_{C2ADF}	30			ns
CDSCLK2 Falling to CDSCLK1 Rising	t_{C2C1}	5			ns
Aperture Delay for CDS Clocks	t_{AD}		2		ns
SERIAL INTERFACE					
Maximum SCLK Frequency	f_{SCLK}	10			MHz
SLOAD to SCLK Set-Up Time	t_{LS}	10			ns
SCLK to SLOAD Hold Time	t_{LH}	10			ns
SDATA to SCLK Rising Set-Up Time	t_{DS}	10			ns
SCLK Rising to SDATA Hold Time	t_{DH}	10			ns
SCLK Falling to SDATA Valid	t_{RDV}	10			ns
DATA OUTPUTS					
Output Delay	t_{OD}		6		ns
3-State to Data Valid	t_{DV}		10		ns
Output Enable High to 3-State	t_{HZ}		10		ns
Latency (Pipeline Delay)			3 (Fixed)		Cycles

8. Vitis软件平台



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



产品

解决方案

资源与支持

商城



处理器

加速器

显卡

自适应 SoC、FPGA 和 SOM >

软件、工具和应用

自适应 SoC 和 FPGA

Versal 产品系列

SoC 产品系列

FPGA 产品系列

成本优化型产品系列

模块化系统 (SOM)

SOM 概述

Kria SOM

KV260 视觉 AI 入门套件

KR260 机器人入门套件

技术

AI 引擎

设计安全性

数字信号处理

功能安全

高速串行

显存解决方案

能效

开发者资源

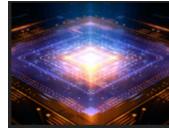
IP

设计中心

开发者中心

客户培训

使用Vivado软件进行开发



16nm

台积电的 16nm FinFET 工艺与全新 UltraRAM 和 SmartConnect 技术相结合，助力 AMD 继续为市场提供“超越摩尔定律”的价值。

AMD
SPARTAN
UltraScale+

AMD
ARTIX
UltraScale+

AMD
KINTEX
UltraScale+

AMD
VIRTEX
UltraScale+



28nm

工艺节点上的持续创新使新器件能够以更低的功耗在整个产品系列中实现最佳性能，以满足关键应用的要求。

AMD
SPARTAN⁷

AMD
ARTIX⁷

AMD
KINTEX⁷

AMD
VIRTEX⁷



20nm

AMD 增强型 FPGA 架构包含步进功能，可为第二代 3D IC 架构提升连接资源数量与相关晶片间带宽。

AMD
KINTEX
UltraScale

AMD
VIRTEX
UltraScale



45nm

AMD 提供卓越的连接功能，例如高逻辑引脚比、小尺寸封装、MicroBlaze™ 软处理器，以及多种受支持的 I/O 协议。

AMD
SPARTAN⁶

8. Vitis软件平台



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



产品

解决方案

资源与支持

商城



处理器

加速器

显卡

自适应 SoC、FPGA 和 SOM >

软件、工具和应用

自适应 SoC 和 FPGA

Versal 产品系列

SoC 产品系列

FPGA 产品系列

成本优化型产品系列

模块化系统 (SOM)

SOM 概述

Kria SOM

KV260 视觉 AI 入门套件

KR260 机器人入门套件

技术

AI 引擎

设计安全性

数字信号处理

功能安全

高速串行

显存解决方案

能效

开发者资源

IP

设计中心

开发者中心

客户培训

Zynq™ 7000 SoC



成本优化的可扩展 SoC 平台

- 单核或双核 Arm Cortex®-A9
- 28nm 7 系列可编程逻辑
- 速率高达 12.5G 的收发器
- 7 系列器件的产品生命周期至少延长至 2035 年

Zynq UltraScale+™ MPSoC



业界首款异构自适应 SoC

- 双核或四核 Arm Cortex-A53
- 双核 Arm Cortex-R5F
- 16nm FinFET+ 可编程逻辑
- Arm Mali™-400MP2
- H.264/H.265 视频编解码器

Zynq UltraScale+ RFSoC



业界首个单芯片自适应无线电平台

- 四核 Arm Cortex-A53
- 双核 Arm Cortex-R5F
- 16nm FinFET+ 可编程逻辑
- 数字 RF-ADC、RF-DAC、SD-FEC
- 可编程的片上网络

Versal™ 自适应 SoC



自适应 SoC

- 双核 Arm Cortex-A72
- 双核 Arm Cortex-R5F
- 7nm 可编程逻辑
- DSP 和 AI 引擎

第二代 Versal 自适应 SoC

第二代 Versal AI Edge 和 Versal Prime 系列为 AI 驱动和经典的嵌入式系统赋予单芯片智能。在性能、功耗、占用面积、功能性安全和安全功能之间实现出色的平衡。与第一代产品系列相比，第二代 Versal AI Edge 和 Versal Prime 系列的子量算力均可提升多达 10 倍。



第二代 AI Edge 系列

采用一体化器件设计，融合新一代 AI 引擎、高性能集成 CPU 和可编程逻辑，面向 AI 驱动的嵌入式系统，支持预处理、AI 推理和后期处理。



第二代 Prime 系列

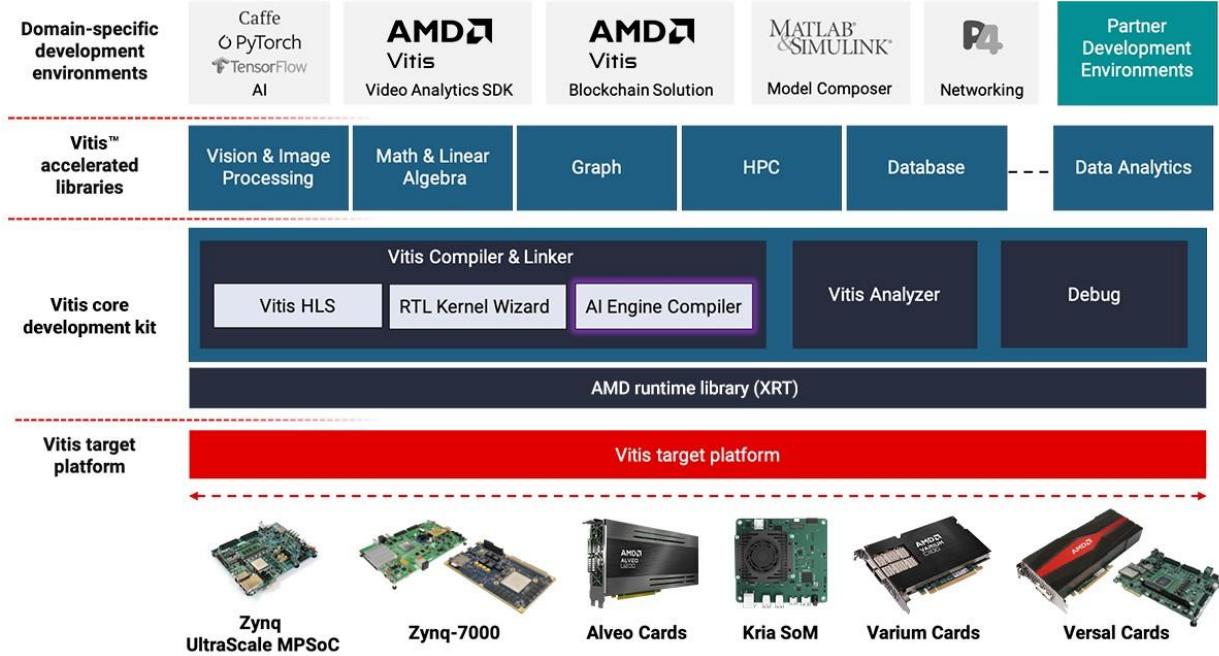
借助高性能集成 CPU、可编程逻辑和 8K 视频处理，打造性能出色的适用于各类市场的经典嵌入式系统。

使用Vitis软件进行开发

8. Vitis软件平台



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



Vitis 软件平台开发环境

AMD Vitis™ 软件平台是一款开发环境，主要用于开发包括 FPGA 架构、Arm® 处理器子系统和 AI 引擎在内的设计。Vitis 工具与 AMD Vivado™ ML Design Suite 相结合，可为设计开发提供更高层次的抽象。

Vitis 软件平台包括以下工具：

- Vitis Embedded - 用于开发在嵌入式 Arm 处理器上运行的 C/C++ 应用代码
- 编译器和仿真器 - 用于使用 AI 引擎阵列实现设计
- Vitis HLS - 用于开发基于 C/C++ 的 IP 块，该 IP 块主要针对 FPGA 架构
- Vitis Model Composer 是一种基于模型的设计工具，可在 MathWorks Simulink® 环境中进行快速设计探索。
- 一系列性能优化的开源库函数，如 DSP、视觉、求解器、超声和 BLAS 等，其可采用 FPGA 架构实现，也可使用 AI 引擎实现

Version

2023.2
2023.1
Vitis 存档
SDSoC 存档
SDAccel 存档
SDK/PetaLinux 存档

Vitis Core 开发套件 - 2023.2 Full Product Installation

重要信息

Vitis™ 统一软件平台 2023.2 版本亮点:

- 最新 Vitis 统一 IDE (GUI) – 该 GUI 在 Vitis、Vitis AIE 编译器/仿真器和 Vitis HLS 之间保持一致
- 全新独立 Vitis 嵌入式软件 (SDK) – 面向为 ARM 处理器子系统 (以及软 Micro Blaze 处理器) 编写嵌入式 C 代码的设计人员
- DSP 设计的 AIE 工具流程增强功能：全新 DSP 库功能，以及在 AIE 仿真器/编译器中为 DSP 功能及新特性提供的最新 API 支持
- 为支持 AIE-ML 的 Versal AI 边缘提供的 Vitis 支持
- 配置文件、调试与跟踪的新特性 – 面向 Versal AIE 器件系列

如需安装 Vitis Core 开发套件，请在统一安装程序上选择 Vitis。Vitis 安装包包括 Vivado™ Design Suite、Vitis Model Composer、Vitis HLS。

无需单独安装 Vivado。

8. Vitis软件平台



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS

[首页](#) / [设计工具](#) / [Vitis 统一软件平台](#) / [Vitis 软件平台](#)

Vitis软件平台

[Vitis 软件平台](#)

[Vitis AI](#)

[Vitis 视频分析
SDK](#)

[Vitis 库](#)

[Vitis HLS](#)

[Vitis Model
Composer](#)



适合从边缘到云的所有开发者

[下载](#)

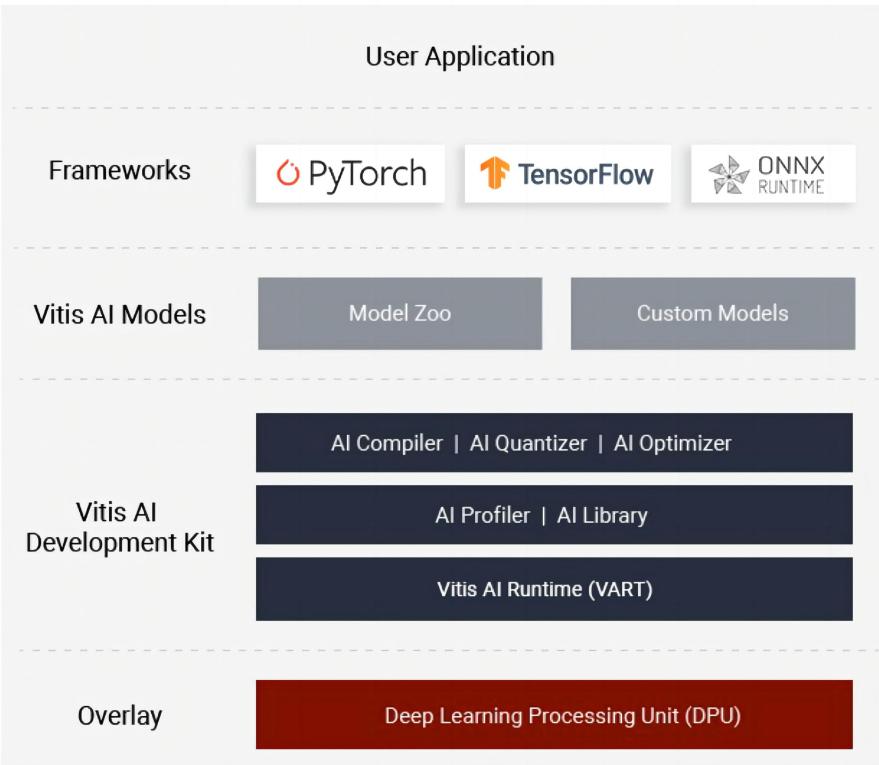
[新增功能](#)

[开发者网站](#)

8. Vitis软件平台



國科大杭州高茅研究院
Hangzhou Institute for Advanced Study, UCAS



Vitis AI

如何使用 Vitis AI 软件进行开发：

- 支持主流框架和最新模型，能够进行多种深度学习推理、CNN 和生成大语言模型
- 功能强大的量化器和优化工具可实现优化的模型精度和处理效率
- 便捷的编译流程和高层次 API 可实现自定义模型的极速部署
- 可配置的高效率 DPU 内核能够充分满足边缘、端点及云端对吞吐量、时延、和功耗的不同需求



边缘部署

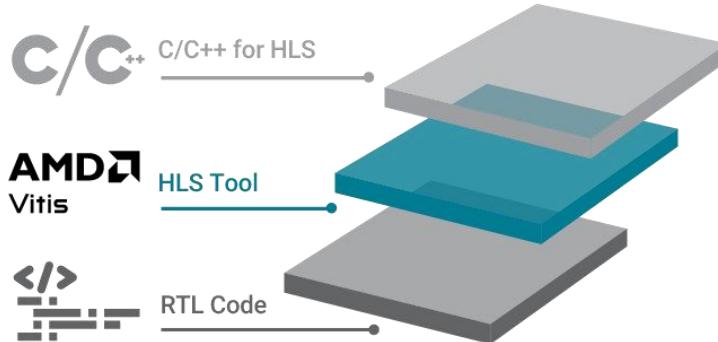
Vitis™ AI 软件不仅可为边缘设备提供支持优化算法的强大计算性能，同时还可凭借优化的功耗实现部署的高度灵活性。它可为汽车、工业、医疗以及视频分析等普及型边缘应用提高计算性能。

[浏览 AMD 及合作伙伴的边缘平台 >](#)

8. Vitis软件平台



Vitis HLS



Vitis™ HLS 工具允许用户通过将 C/C++ 函数综合成 RTL，轻松创建复杂的 FPGA 算法。Vitis HLS 工具与 Vivado™ Design Suite（用于综合、布置和布线）及 Vitis™ 统一软件平台（用于所有异构系统设计和应用）高度集成。

- 使用 Vitis HLS 流程，用户可针对 C 代码应用指令，创建专门用于所需实现方案的 RTL。
- 不仅可从 C 语言源代码创建多个设计架构，而且还可启用用于高质量 Correct-by-Construction RTL 的路径。
- C 语言仿真可用于验证设计，支持比基于 RTL 的传统仿真更快的迭代。
- Vitis HLS 工具具有一系列丰富的分析及调试工具，其可促进设计优化。

C 至 RTL 的转换

Vitis HLS 工具将对 C 语言代码的不同部分执行不同的综合：

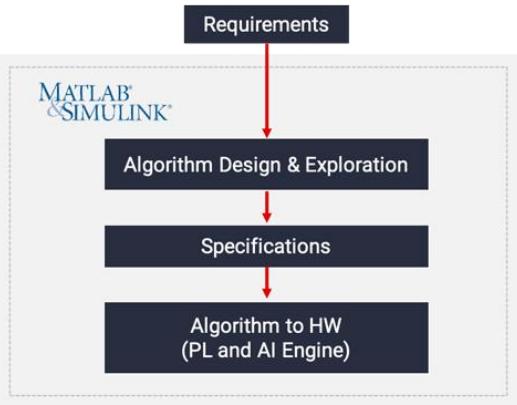
-
- ```
graph TD; C[C/C++]; D[Constraints & Devices]; C --> S[C Synthesis]; D --> S; S --> RTL[RTL]; subgraph Box []; direction LR; T1[Top Level Function] --- L1[Loops]; OCF[Other C Functions] --- A[Arrays]; end; Box --> S;
```
- C/C++ 代码的顶层函数参数不仅可综合成 RTL I/O 端口，而且还可通过接口综合硬件协议自动执行。
  - 其它 C 语言函数将综合至 RTL 模块中，保持设计层级。
  - C 语言函数循环保持滚动或流水线，以提高性能。
  - C 语言代码阵列可指向任何内存资源，如 BRAM、LUTRAM 和 URAM 等。
  - 时延、初始化间隔、循环迭代时延和资源利用率等性能指标可使用综合报告查看。
  - Vitis HLS 工具的编译指示及优化指令允许配置 C/C++ 代码的综合结果。

# 8. Vitis软件平台



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

## Vitis Model Composer



### Vitis Model Composer 概述

Vitis™ Model Composer 是一个基于模型的设计工具，不仅可在 MathWorks MATLAB® 和 Simulink® 环境中进行快速设计探索，而且还可通过自动代码生成在 AMD 器件上加速投产进程。

您可以设计您的 DSP 算法并使用高层次性能优化模块对其进行迭代，同时还可通过系统级仿真验证功能正确性。Vitis Model Composer 可通过自动优化将您的设计转换为生产质量级实现方案。

该工具提供一个具有 200 多个 HDL、HLS 和 AI 引擎模块的库，用于在 AMD 器件上设计并执行算法。此外，它还允许将自定义 HDL、HLS 和 AI 引擎代码按模块导入工具。

Vitis Model Composer 提供了一种设计 Versal™ 自适应 SoC 的方法。该工具可以帮助设计者创建针对 PL 和 AI 引擎阵列的复杂系统。了解如何将 Versal AI 引擎与 Vitis Model Composer 结合使用。

Vitis Model Composer 为 DSP 提供 AMD 系统生成器的所有功能性，自 2021.1 版起，该系统生成器不再按独立工具提供。

# 9.FPGA高级应用



Zynq™ 7000 SoC



成本优化的可扩展 SoC 平台

- 单核或双核 Arm Cortex®-A9
- 28nm 7 系列可编程逻辑
- 速率高达 12.5G 的收发器
- 7 系列器件的产品生命周期至少延长至 2035 年

Zynq UltraScale+™  
MPSoC



业界首款异构自适应 SoC

- 双核或四核 Arm Cortex-A53
- 双核 Arm Cortex-R5F
- 16nm FinFET+ 可编程逻辑
- Arm Mali™-400MP2
- H.264/H.265 视频编解码器

Zynq 7000 SoC 器件

Zynq UltraScale+ RFSoC



业界首个单芯片自适应无线电平台

- 四核 Arm Cortex-A53
- 双核 Arm Cortex-R5F
- 16nm FinFET+ 可编程逻辑
- 数字 RF-ADC、RF-DAC、SD-FEC

Zynq UltraScale+ MPSoC 器件

Zynq UltraScale+ RFSoC 器件

**ZYNQ**

# 9.FPGA高级应用



## 特性简介

### 采用 CoreSight™ 技术的双核 ARM Cortex-A9

无与伦比的性能功耗比

- 选择 ARM Cortex-A9 处理器为常见应用产品实现最佳性能功耗比
- 支持单精度和双精度浮点
- 运行速率高达 1GHz

### 最大型超高性能存储器系统

配备业界最快速的存储器控制器和最大容量的片上存储器

- 512KB L2 高速缓存
- 256KB 片上存储器可容纳整个实时操作系统
- 集成存储器控制器支持 DDR3-1866

### 7 系列 28nm 可编程逻辑

HPL 工艺实现最佳的性能功耗比

- Artix®-7 FPGA 架构实现低功耗和低成本
- Kintex®-7 FPGA 架构实现最佳的性能 / 价格 / 功耗比

### 集成存储器映射外设

采用常用协议

- 2x USB 2.0 (OTG), 带有 DMA
- 2x Tri-mode 千兆位以太网, 带有 DMA
- 2x SD/SDIO, 带有 DMA
- 2x UART、2x CAN 2.0B、2x I2C、2x SPI、32b GPIO

### All Programmable 电源管理

处理系统和可编程逻辑中的多种电源优化技术

- 灵活可调的功率范围实现可调节的处理器、互连和存储器速度
- ARM 低功耗模式
- 部分重配置可降低可编程逻辑要求

### AMBA 开放标准互连端口

处理系统与可编程逻辑之间的高带宽互连

- 64 位 AXI ACP 端口为附加的软处理器实现增强的硬件加速性能和缓存一致性
- PS 与 PL 之间带宽高达 100Gb/s

### 大量并行信号处理

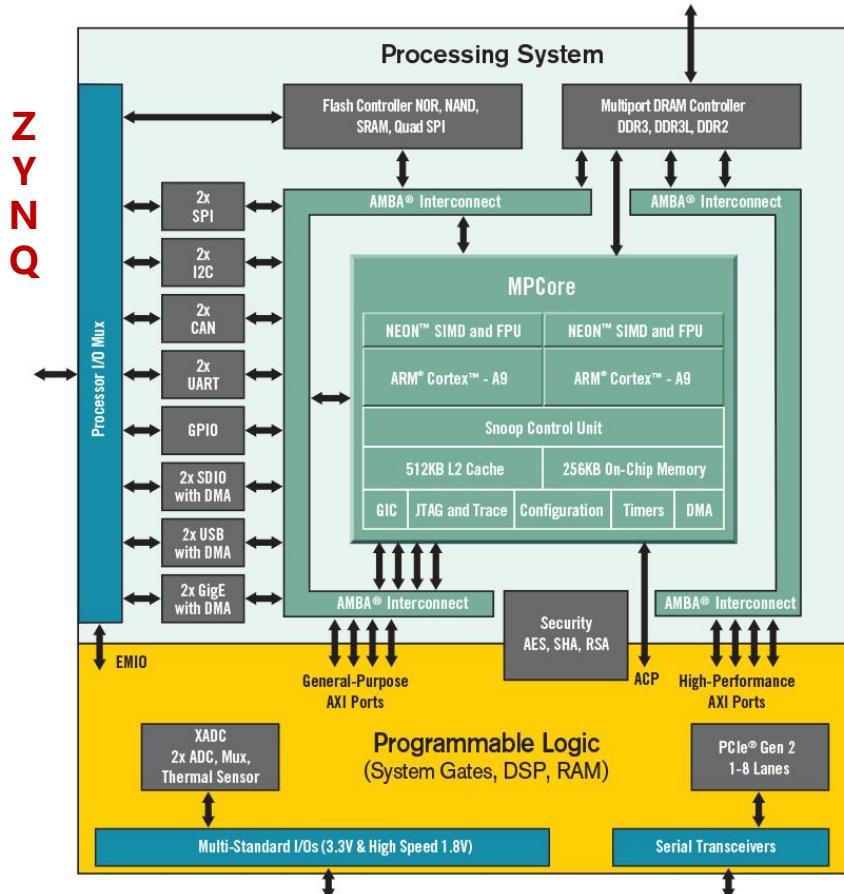
实现包括视频处理和分析在内的硬件加速

- 专用的完全定制的低功耗 DSP Slice
- 多达 2,020 个 DSP 模块, 可提供超过 2,662 GMAC 的性能

### 高保密性、安全性和可靠性

针对硬件、安全系统引导和软件执行的最先进技术

- 利用带有安全 ROM 码的片上存储器实现处理器优先引导
- 防篡改 (AT) 技术在检测到篡改活动时将器件“归零”
- 采用基于 RSA 认证、AES-256 解密和 SHA-256 数据认证实现安全系统引导
- 全方位 ARM TrustZone® 支持



# 9.FPGA高级应用

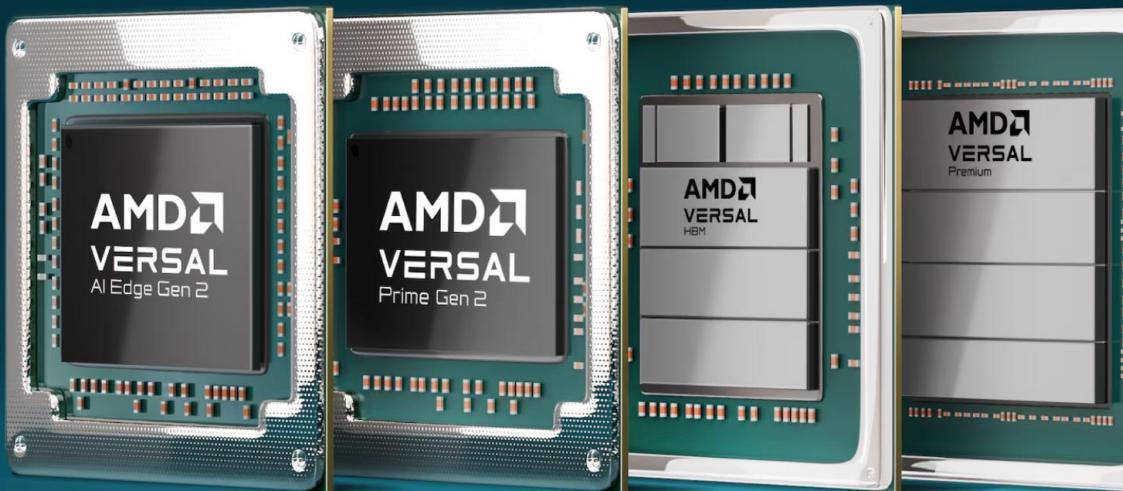


國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study,UCAS

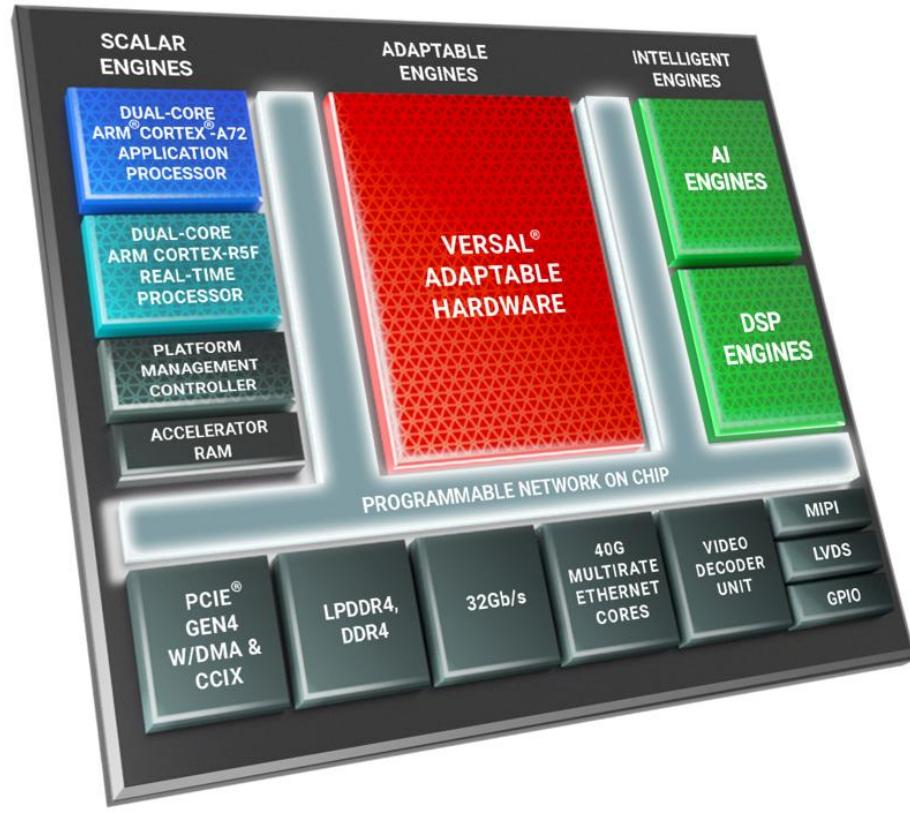
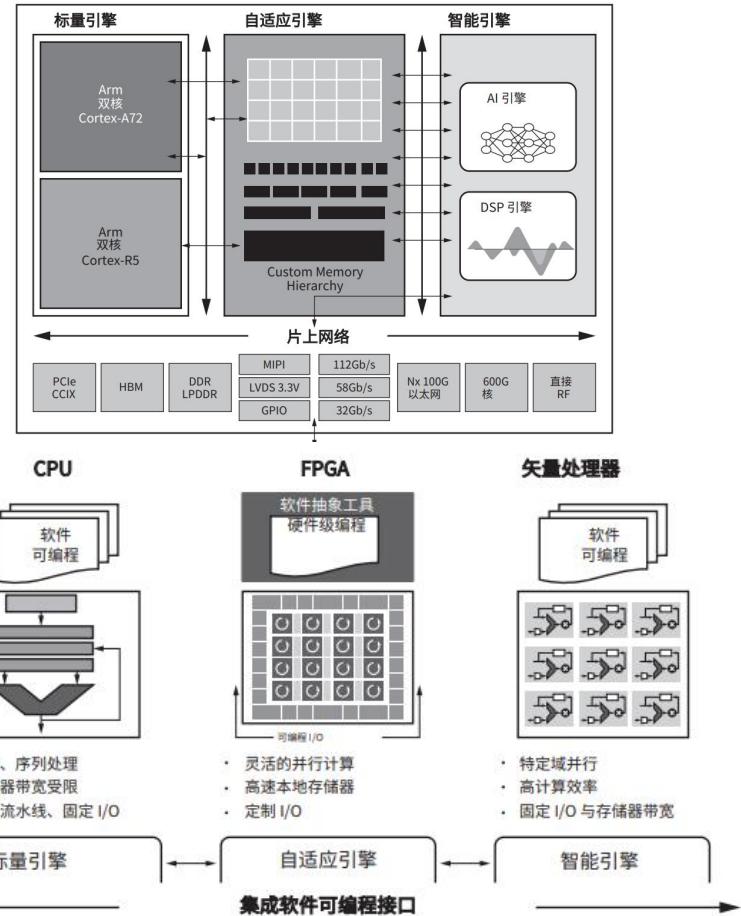
## AMD Versal™ 自适应 SoC

面向从云端到边缘的各类应用

全新第二代 Versal 产品系列



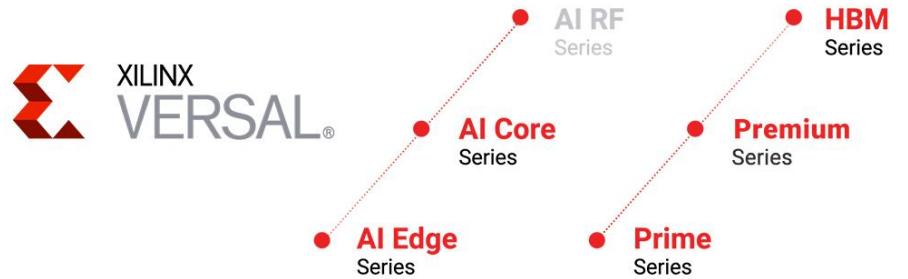
# 9.FPGA高级应用





# 9.FPGA高级应用

ACAP：面向从云到边缘的任何应用



Adaptable for Any Application

Application-Specific Frameworks  
TensorFlow | PyTorch | Caffe

Vitis™ Unified SW Development Environment | Vivado® Design Suite

|                        |           |                                 |                     |
|------------------------|-----------|---------------------------------|---------------------|
| OS & Embedded Run-Time | Custom HW | Xilinx & Ecosystem HW Libraries | C, Xilinx Libraries |
|------------------------|-----------|---------------------------------|---------------------|

|                |                   |                     |
|----------------|-------------------|---------------------|
| Scalar Engines | Adaptable Engines | Intelligent Engines |
|----------------|-------------------|---------------------|

VERSAL ACAP

Software Programmable

Heterogeneous Platform

| 市场     | 基准               | 与 CPU 对比 | 与 GPU 对比 | 与 FPGA 对比 | 注释                                                  |
|--------|------------------|----------|----------|-----------|-----------------------------------------------------|
| 数据中心   | 图像识别（推断）——时延敏感   | 43 倍     | 2 倍      | 5 倍       | GoogLeNet v1 (不限制批处理大小)                             |
|        | 图像识别（推理）——2ms 时延 | 不适用      | 8 倍      | 5 倍       | GoogLeNet v1 (< 2 ms)<br>CPU 时延下限 5ms               |
|        | 风险分析             | 89 倍     | 不适用      | >1 倍      | 用于利率互换 Maxeler 结果的风险价值 (VaR)                        |
|        | 基因组学             | 90 倍     | 不适用      | >1 倍      | 人类基因分析 Edico 基因组结果                                  |
|        | 弹性搜索             | 91 倍     | 不适用      | >1 倍      | 1TB 数据 BlackLynx 结果时延降低 91 倍                        |
| 无线 5G  | 16x16 5G 远程无线电   | 不适用      | 不适用      | >5 倍      | 为 5G 远程无线电提供 >5 倍的无线电带宽                             |
|        | 波束形成             | 不适用      | 不适用      | >5 倍      | >5 倍的计算能力                                           |
| A&D 雷达 | DSP TMACs        | 不适用      | 不适用      | >5 倍      | 超过 27 TMAC                                          |
|        | 算法迭代时间           | 不适用      | 不适用      | >100 倍    | 软件可编程智能引擎在几分钟内编译完毕                                  |
| 汽车     | 低时延推断 (<2 ms)    | 不适用      | 3 倍      | 15 倍      | ResNet50 Batch=1<br>AI 引擎能更好地适应低时延、安全关键型 ADAS 和自动驾驶 |
|        | 外壳类型             | 1        | 2        | 4         | ACAP 产品组合是唯一能够高效支持 <10W、20W、30W，以及后备箱安装外壳的器件        |
| 有线     | 加密网络流量           | 不适用      | 不适用      | 4 倍       | ACAP 对网络和加密 IP 的集成使多太比特的单芯片实现成为可能。                  |

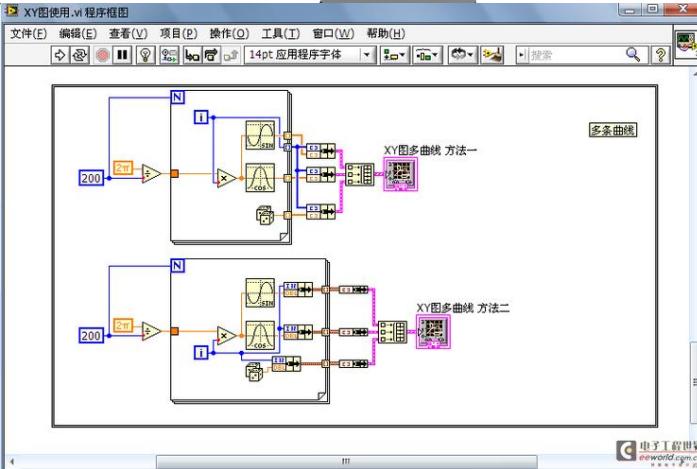
# 9.FPGA高级应用



## ■ 什么是LabVIEW

- LabVIEW是一种**图形化编程环境**, 工程师可使用该环境来开发自动化研究、验证和生产测试系统
- LabVIEW是一种**类似 C 语言一样的编程语言**
- 不仅可以实现在 windows、linux 和 mac OS 上的程序开发, 也能直接开发传统的嵌入式芯片, 例如: 单片机、DSP、ARM、FPGA 等
- LabVIEW采用的是**数据流模型**, 而不是顺序文本代码行, 可以减少花在语句和语法的时间

```
6 #include <stdio.h>
7 #define PI 3.14
8
9 double area(int *pRadius, int *pHeight);
10
11 int main(void)
12 {
13 int r, h;
14 double dArea;
15 printf("input radius of cylinder: ");
16 scanf("%d", &r);
17 printf("input height of cylinder: ");
18 scanf("%d", &h);
19
20 dArea = area(&r, &h);
21 printf("area of cylinder: %lf", dArea);
22
23 return 0;
24 }
25
26 double area(int *pRadius, int *pHeight)
27 {
28 double dPerimeter;
29 *pRadius /= 2;
30 *pHeight *= 2;
31 dPerimeter = 2 * PI * (*pRadius);
32
33 return PI * (*pRadius * *pRadius) * 2 + dPerimeter * (*pHeight);
34 }
```



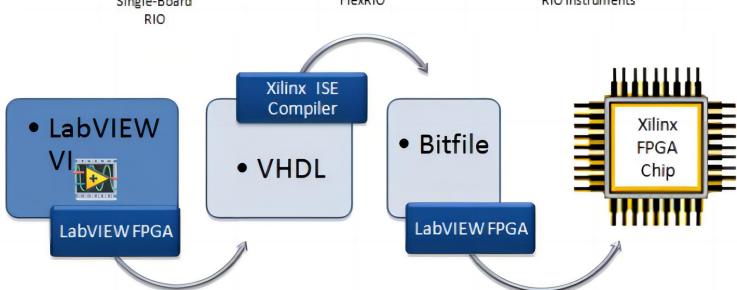
# 9.FPGA高级应用



## ■ LabVIEW开发FPGA的优缺点

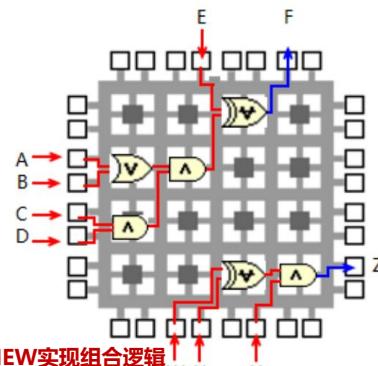
1. LabVIEW语言学习简单，无需掌握Verilog/VHDL等硬件描述语言
2. 程序设计简单，可以更关注顶层实现（对入门级使用者友好）
3. LabVIEW采用数据流驱动方式，与硬件执行方式一致
4. 带有高级处理功能，包括PID、信号处理等模块
5. LabVIEW开发FPGA，实现相同功能所占的资源多
6. 开发平台较为封闭，大部分需要基于NI的硬件
7. 国产平台：芒果树、神电测控

## LabVIEW FPGA Targets



User Generated  
Auto Generated

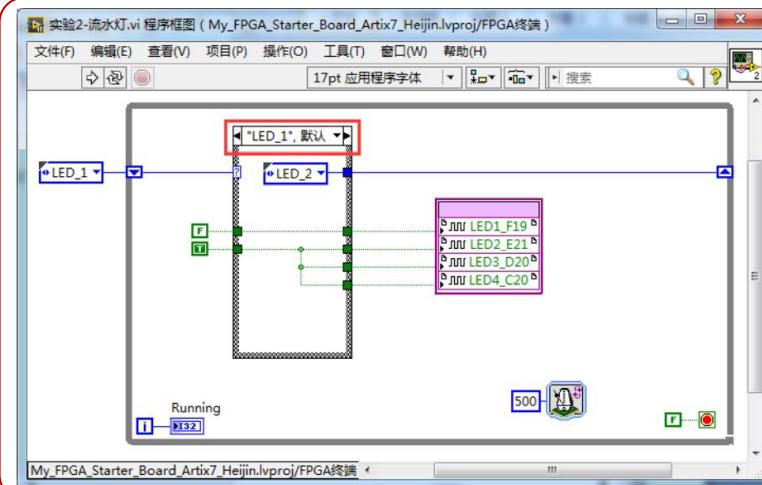
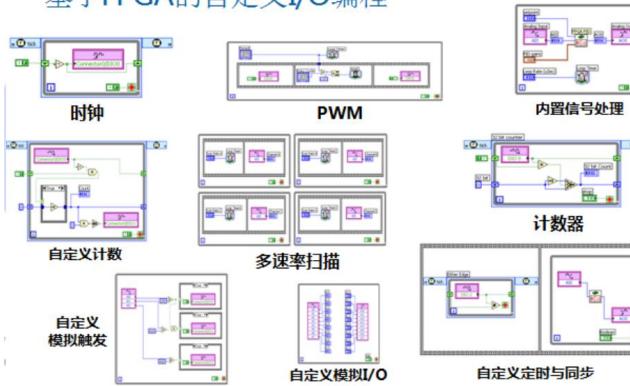
# 9.FPGA高级应用



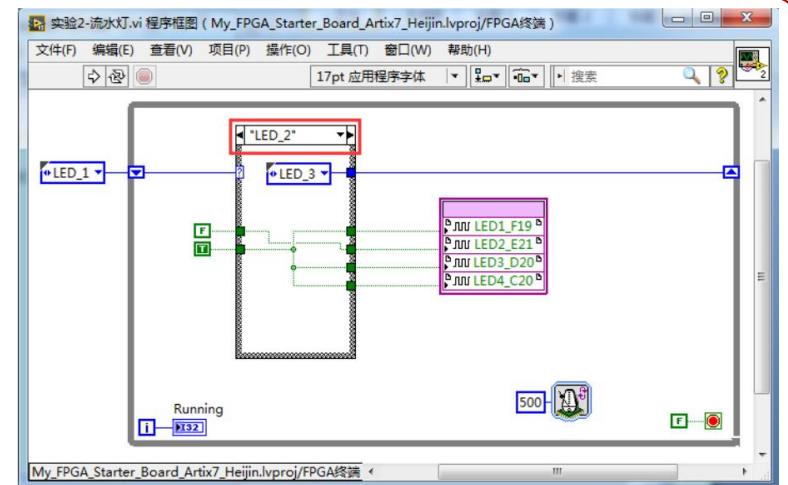
LabVIEW实现组合逻辑

$$F = \{(A+B)CD\} \oplus E$$

基于FPGA的自定义I/O编程



LabVIEW实现状态机





# THANKS



國科大杭州高等研究院  
Hangzhou Institute for Advanced Study, UCAS



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS



# FPGA电路软硬件设计 第十一讲 (11.2 Lab9串口发送实验)

2025年5月13日

# UART串口通讯



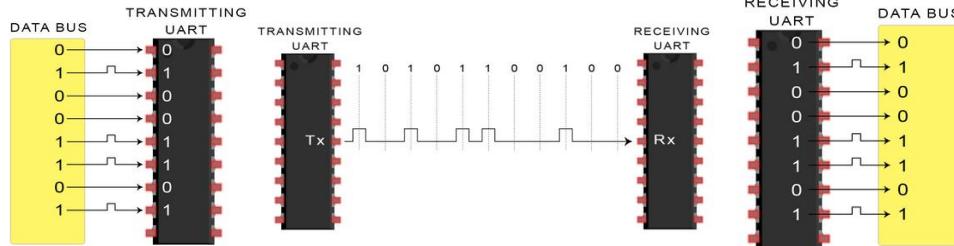
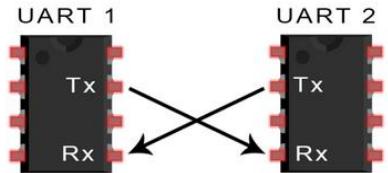
國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

## ■什么是UART

- 全称是通用异步收发传输器 (Universal Asynchronous Receiver/Transmitter)
- 把并行输入信号转成串行输出信号的通讯协议
- 一种通用串行数据总线, 用于异步通信。
- 该总线为双向通信, 可以实现全双工传输和接收

## ■通讯协议

- 空闲位**: 当总线处于空闲状态时信号线的状态为 '1' , 即高电平
- 起始位**: 开始进行数据传输时发送方要先发出一个低电平' 0' 来表示传输字符的开始
- 数据位**: 起始位之后就是要传输的数据, 数据可以是5, 6, 7, 8, 9位, 构成一个字符, 一般都是8位。先发送最低位最后发送最高位。
- 奇偶校验位**: 分为无校验、奇检验 (数据位+校验位合计为单数个1) 、偶校验
- 停止位**: 数据结束标志, 可以是1位、1.5位、2位的高电平
- 波特率**: 数据传输速率使用波特率来表示, 常见的有9600bps, 115200bps, 传输一个bit需要的时间为[1/波特率(秒)]





## ■ UART串口通讯举例

- 以9600 8N1 (9600波特率，8个数据位，没有校验位，1位停止位) 为例
- 传输' O' 'K' 两个ASCII值
- 'O' 的ASCII为0x4F，对应的二进制数据为01001111
- 'K' 的ASCII为0x4B，对应的二进制数据为01001011
- 传输方式为从最低位 (LSB) 开始传输
- 串口波特率为9600，1bit传输时间为104us，传送一个数据实际上是10个比特（起始位，8个数据位，停止位），一个byte的传输速率为 $9600 \times 8 / 10 = 7680 \text{ bps}$



# FIFO队列



## ■什么是FIFO

- FIFO (First In First Out) 队列是一种数据缓冲器，用于数据的缓存，可以跨时钟域使用
- 是一种先入先出的存储器，即最先写入的数据，最先读出
- FIFO的参数有数据深度和数据宽度，数据宽度是指存储数据的宽度，深度是指存储器可以存储多少个数据

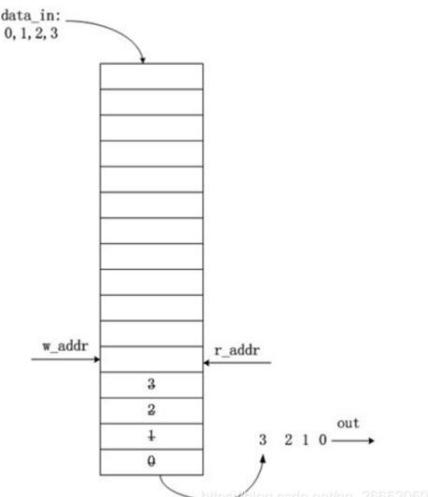


## ■ FIFO的标志位

- FIFO队列有两个主要标志位，一个满和一个空标志位，分别标志FIFO数据写满和数据读空
- 在数据写满状态下，数据写入是不允许的，因此在这个状态下，写入的数据无效
- 而数据读空状态下，数据读取是不允许的，因此在这个状态下，读取的数据无效

## ■ FIFO的位置指针

- FIFO队列有两个位置指针，一个是写指针，指向队列的第一个存储单元，一个读指针，指向队列的最后一个存储单元
- 当有写命令的时候，数据写入写指针指向的存储单元，然后写指针加一
- 当有读命令的时候，读指针加一，再读出读指针指向的存储单元的数据
- 可以通过Vivado的FIFO IP核实现配置及管理



# FIFO队列



## ■ Vivado的FIFO IP核

- Common Clock模式: 同步FIFO (即输入输出信号的时钟相同)
- Independent Clock模式: 异步FIFO (即输入输出信号的时钟不同)
- 物理实现方式: Block RAM、Distributed RAM、Shift Register、Builtin FIFO

The screenshot shows the Vivado IP Catalog interface. A search bar at the top contains the text "Search: fifo". Below it, a table lists search results. The first result is "AXI Data FIFO" under the "Name" column, with "AXI4" in the "Type" column, "Production" in "Status", "Included" in "License", and "xilinx.com:ip:axi\_data\_fifo:2.1" in "VLAN". The second result is "FIFO Generator" under the "Name" column, with "AXI4, AXI4-Stream" in the "Type" column, "Production" in "Status", "Included" in "License", and "xilinx.com:ip:fifo\_generator:13.2" in "VLAN". The "FIFO Generator" row is highlighted with a blue background.

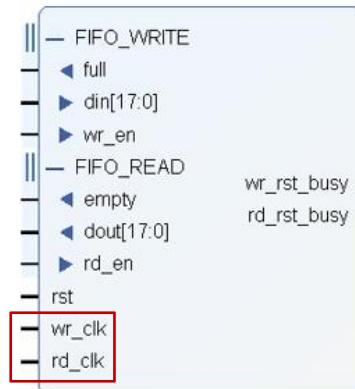
| Name                        | Type              | Status     | License  | VLAN                              |
|-----------------------------|-------------------|------------|----------|-----------------------------------|
| AXI Data FIFO               | AXI4              | Production | Included | xilinx.com:ip:axi_data_fifo:2.1   |
| FIFO Generator              | AXI4, AXI4-Stream | Production | Included | xilinx.com:ip:fifo_generator:13.2 |
| Memories & Storage Elements |                   |            |          |                                   |
| Video & Image Processing    |                   |            |          |                                   |

**Details**

Name: FIFO Generator  
Version: 13.2 (Rev. 4)  
Interfaces: AXI4, AXI4-Stream



Common Clock模式  
相同时钟域



Independent Clock模式  
跨时钟域

# FIFO队列



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

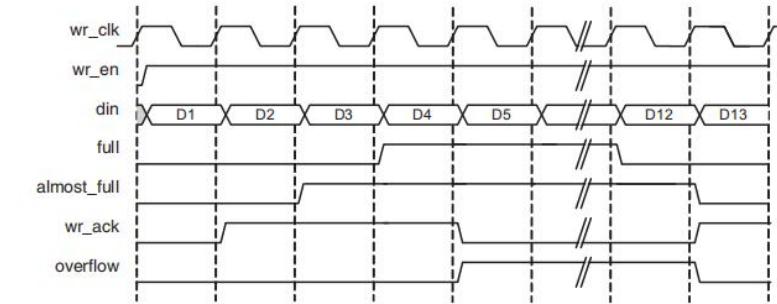
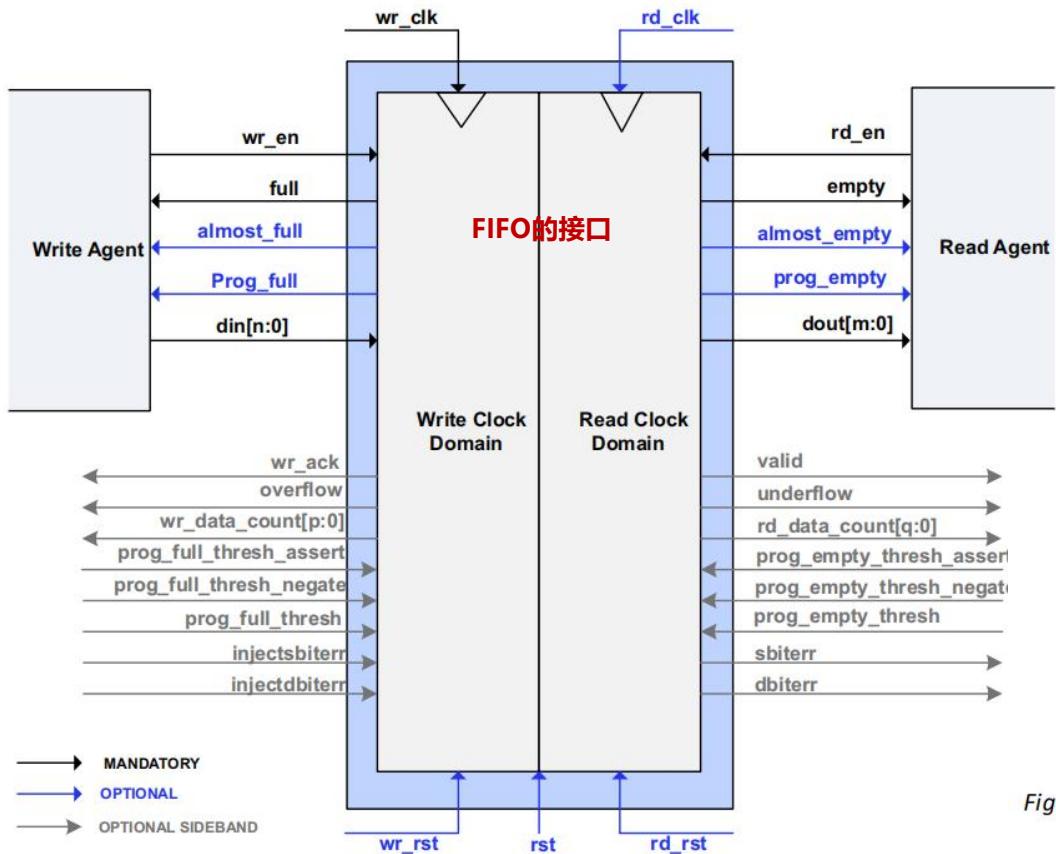


Figure 3-2: Write Operation for a FIFO with Independent Clocks  
FIFO写入时序: wr\_en高电平, 通过din写入数据, 关注full标志

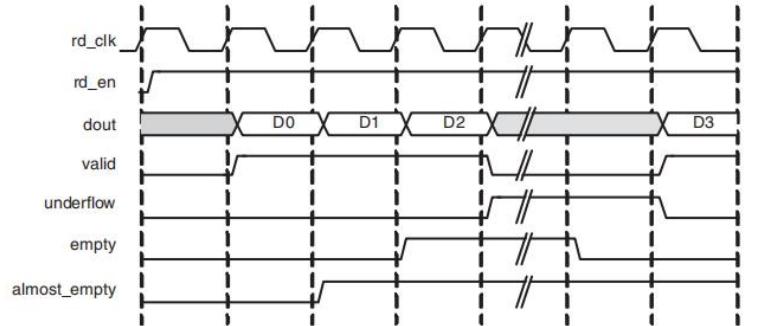


Figure 3-3: Standard Read Operation for a FIFO with Independent Clocks  
FIFO读出时序: rd\_en高电平, 通过dout读出数据, 关注empty标志

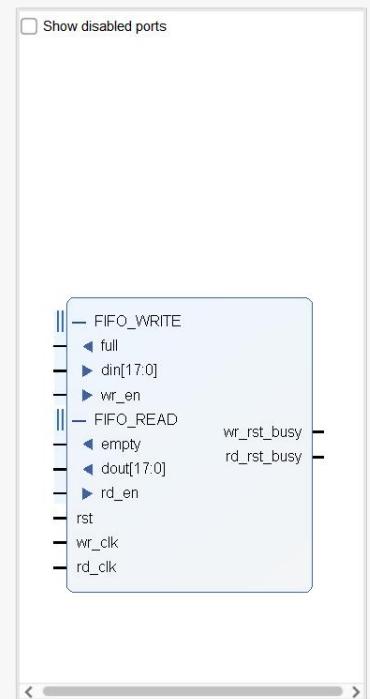
# FIFO队列



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

## FIFO Generator (13.2)

[Documentation](#) [IP Location](#) [Switch to Defaults](#)



Component Name: fifo\_generator\_0

Basic Native Ports Status Flags Data Counts Summary

Interface Type

接口  Native  AXI Memory Mapped  AXI Stream

Fifo Implementation: Independent Clocks Block RAM

Synchronization Stages: Common Clock Block RAM, Common Clock Distributed RAM, Common Clock Shift Register

FIFO Implementation Configuration:

| Supported Features                  | Memory Type     | (1) | (2) | (3) | (4) | (5) |
|-------------------------------------|-----------------|-----|-----|-----|-----|-----|
| Common Clock (CL)                   | SRAM            | ✓   | ✓   |     | ✓   | ✓   |
| Common Clock (CL)                   | Built-in RAM    | ✓   |     |     |     |     |
| Independent Clocks (RD_CLK, WR_CLK) | Block RAM       | ✓   | ✓   | ✓   | ✓   |     |
| Independent Clocks (RD_CLK, WR_CLK) | Distributed RAM | ✓   |     |     |     |     |
| Independent Clocks (RD_CLK, WR_CLK) | Built-in FIFO   | ✓   | ✓   | ✓   | ✓   |     |

(1) Non-symmetric aspect ratios (different read and write data widths)  
(2) First-Word Fall-Through  
(3) Uses Built-in FIFO primitives  
(4) ECC support  
(5) Dynamic Error Injection

能够支持的列表及说明

## FIFO Generator v13.2

## LogiCORE IP Product Guide

Vivado Design Suite

PG057 October 4, 2017

建议阅读一遍使用文档

# FIFO队列



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

Customize IP

FIFO Generator (13.2)

Documentation IP Location Switch to Defaults

Show disabled ports

Component Name: fifo\_generator\_0

Basic Native Ports Status Flags Data Counts Summary

**Read Mode**

Standard FIFO  First Word Fall Through

读出模式选择  
标准模式：数据读出延迟一个时钟周期  
FWFT模式：读出无延迟

**Data Port Parameters**

Write Width: 8 (1,2,3..1024) 写入数据位宽

Write Depth: 1024 数据深度 Actual Write Depth: 1023

Read Width: 8 读出数据位宽，可以与写入位宽不一致

Read Depth: 1024 Actual Read Depth: 1023

**ECC, Output Register and Power Gating Options**

ECC Hard ECC Single Bit Error Injection Double Bit Error Injection

ECC Pipeline Reg Dynamic Power Gating

Output Registers Embedded Registers

**Initialization**

Reset Pin  Enable Reset Synchronization  Enable Safety Circuit

Reset Type: Asynchronous Reset

Full Clean Default Value

+ FIFO\_WRITE  
+ FIFO\_READ  
rst  
wr\_clk  
rd\_clk

wr\_rst\_busy  
rd\_rst\_busy

Component Name: data\_fifo

Basic Native Ports Status Flags Data Counts Summary

**WARNING :** The use of Asynchronous Reset can lead to BRAM data corruption(AR 42571). It is recommended to Enable Safety Circuit

Block RAM resource(s) (18K BRAMs): 1  
Block RAM resource(s) (36K BRAMs): 0

|                                               |                              |
|-----------------------------------------------|------------------------------|
| Clocking Scheme                               | Independent Clocks           |
| Memory Type                                   | Block RAM                    |
| Model Generated                               | Behavioral Model             |
| Write Width                                   | 8                            |
| Write Depth                                   | 256                          |
| Read Width                                    | 8                            |
| Read Depth                                    | 256                          |
| Almost Full/Empty Flags                       | Not Selected/Not Selected    |
| Programmable Full/Empty Flags                 | Not Selected/Not Selected    |
| Data Count Outputs                            | Not Selected                 |
| Handshaking                                   | Not Selected                 |
| Read Mode / Reset                             | Standard FIFO / Asynchronous |
| Read Latency (From Rising Edge of Read Clock) | 1                            |

**标准模式**

Component Name: data\_fifo

Basic Native Ports Status Flags Data Counts Summary

**WARNING :** The use of Asynchronous Reset can lead to BRAM data corruption(AR 42571). It is recommended to Enable Safety Circuit

Block RAM resource(s) (18K BRAMs): 1  
Block RAM resource(s) (36K BRAMs): 0

|                                               |                                        |
|-----------------------------------------------|----------------------------------------|
| Clocking Scheme                               | Independent Clocks                     |
| Memory Type                                   | Block RAM                              |
| Model Generated                               | Behavioral Model                       |
| Write Width                                   | 8                                      |
| Write Depth                                   | 257                                    |
| Read Width                                    | 8                                      |
| Read Depth                                    | 257                                    |
| Almost Full/Empty Flags                       | Not Selected/Not Selected              |
| Programmable Full/Empty Flags                 | Not Selected/Not Selected              |
| Data Count Outputs                            | Not Selected                           |
| Handshaking                                   | Not Selected                           |
| Read Mode / Reset                             | First-word Fall-through / Asynchronous |
| Read Latency (From Rising Edge of Read Clock) | 0                                      |

**FWFT模式**

# FIFO队列



FIFO Generator (13.2)

Documentation IP Location Switch to Defaults

Show disabled ports

Component Name: fifo\_generator\_0

Basic Native Ports Status Flags Data Counts Summary

**Optional Flags**

Almost Full Flag  Almost Empty Flag 状态标志选择 根据实际需要进行设置

**Handshaking Options**

**Write Port Handshaking**

Write Acknowledge Active High  Overflow Active High

**Read Port Handshaking**

Valid Flag Active High  Underflow Flag Active High

**Programmable Flags**

Programmable Full Type: No Programmable Full Threshold

Full Threshold Assert Value: 1021 [4 - 1021]

Full Threshold Negate Value: 1020 [3 - 1020]

Programmable Empty Type: No Programmable Empty Threshold

Empty Threshold Assert Value: 2 [2 - 1019]

Empty Threshold Negate Value: 3 [3 - 1020]

+ FIFO\_WRITE  
+ FIFO\_READ  
rst  
wr\_clk  
rd\_clk  
wr\_rst\_busy  
rd\_rst\_busy

# FIFO队列



**FIFO Generator (13.2)**

Documentation IP Location Switch to Defaults

Show disabled ports

Component Name: fifo\_generator\_0

Basic Native Ports Status Flags Data Counts Summary

Data Count Options

More Accurate Data Counts

Data Count

Data Count Width: 10 [1 - 10]

Write Data Count (Synchronized with Write Clk) **写入数据计数器**

Write Data Count Width: 10 [1 - 10]

Read Data Count (Synchronized with Read Clk) **读出数据计数器**

Read Data Count Width: 10 [1 - 10]

+ FIFO\_WRITE  
+ FIFO\_READ  
rst wr\_clk rd\_clk wr\_rst\_busy rd\_rst\_busy

根据实际需要进行设置  
使用时参考IP核产品手册

# Lab9串口发送实验

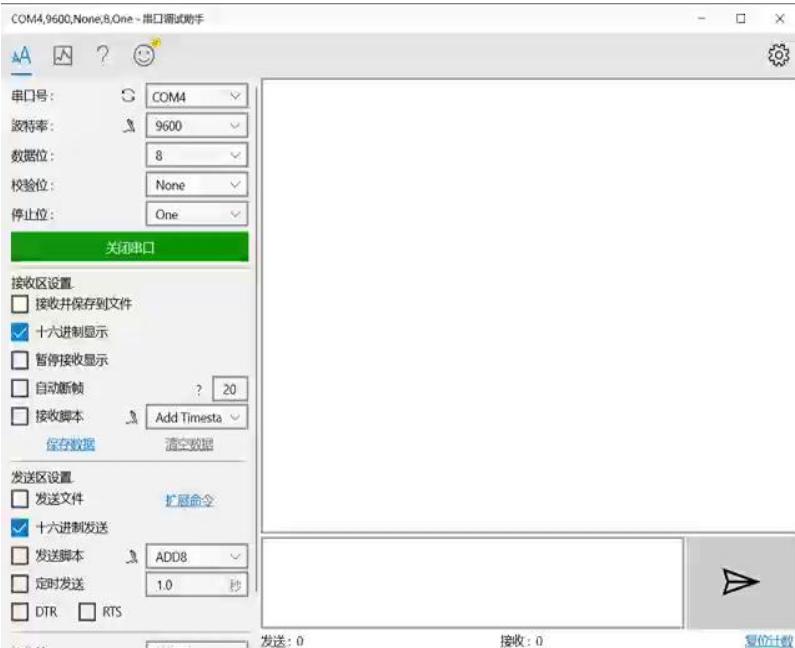


## ■ 实验内容

- 使用FPGA串口输出学号并通过上位机接收显示
- 学号数据产生模块循环产生学号数据，存入到FIFO中
- 串口模块根据FIFO的状态（非空），读取数据后串行发送
- 上位机接收串口循环输出的学号数据

## ■ 实验目的

- 学会使用状态机，并将其应用于数据发生器、串口控制器
- 熟悉FIFO及IP核的使用
- 熟悉UART的时序及使用

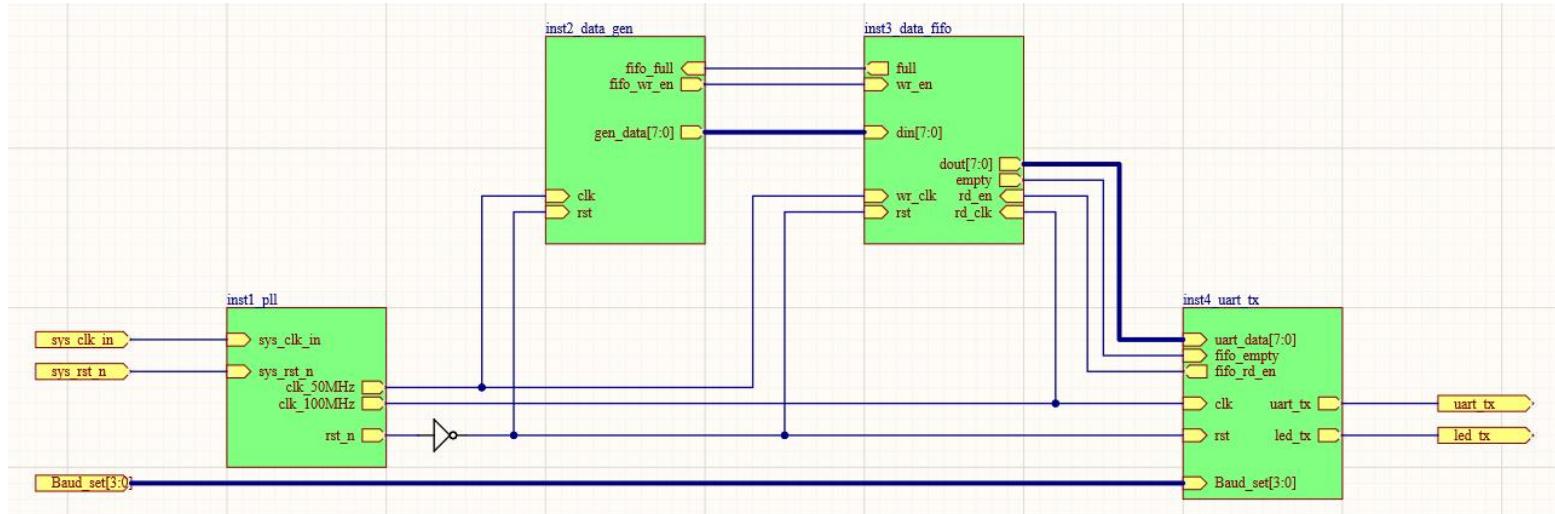


串口调试助手：可以在Micrsoft Store中搜索串口调试助手下载后使用

# Lab9串口发送实验



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS



## ■ 电路结构

- 系统包括四个模块，时钟管理模块、数据产生模块、FIFO模块、串口发送模块
- 时钟模块产生50MHz及100MHz时钟，通过FIFO实现跨时钟域数据传输
- 数据产生模块中使用状态机产生学号数据序列，根据FIFO的满标志状态，将产生的数据写入FIFO
- FIFO模块使用Vivado的IP核产生，设置为异步FIFO，数据8位，数据深度为512
- 串口发送模块产生波特率时钟，使用状态机判断FIFO空标志，将从FIFO读出的并行数据转化为串行输出

# Lab9串口发送实验



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study,UCAS

输入输出端口表

| 序号 | 信号名称          | 信号类型 | 输入/输出 | 信号描述     | FPGA引脚      | 说明         |
|----|---------------|------|-------|----------|-------------|------------|
| 1  | sys_clk_in    | 1bit | 输入    | 系统时钟输入   | P17         | 100MHz输入   |
| 2  | sys_RST_n     | 1bit | 输入    | 系统复位输入   | P15         | 低电平复位      |
| 3  | uart_tx       | 1bit | 输出    | 串行输出     | T4          | 串口输出引脚     |
| 4  | Baud_set[3:0] | 4bit | 输入    | 选择串口的波特率 | R2/M4/N4/R1 | 选择串口的波特率   |
| 5  | led_tx        | 1bit | 输出    | 串口正在传输信号 | K2          | 数据发送时点亮led |

# Lab9串口发送实验



```
module Lab9_UART_TX_Top(
 input sys_clk_in,
 input sys_rst_n,
 input [3:0] Baud_set,
 output uart_tx,
 output led_tx
);

wire clk_50MHz;
wire clk_100MHz;
wire rst;
wire fifo_full;
wire fifo_empty;
wire fifo_wr_en;
wire fifo_rd_en;
wire [7:0] gen_data;
wire [7:0] uart_data;

assign rst = ~rst_n;

/////////////////////////////
clk_pll inst1_clk_pll
(
 .clk_50MHz(clk_50MHz),
 .clk_100MHz(clk_100MHz),
 .resetn(sys_rst_n),
 .rst_n(rst_n),
 .sys_clk_in(sys_clk_in)
);

/////////////////////////////
data_gen inst2_data_gen
(
 .clk(clk_50MHz),
 .rst(rst),
 .full(fifo_full),
 .wr_en(fifo_wr_en),
 .gen_data(gen_data)
);

```

```
/////////////////////////////
data_fifo inst3_data_fifo
(
 .rst(rst),
 .wr_clk(clk_50MHz),
 .rd_clk(clk_100MHz),
 .din(gen_data),
 .wr_en(fifo_wr_en),
 .rd_en(fifo_rd_en),
 .dout(uart_data),
 .full(fifo_full),
 .empty(fifo_empty)
);

/////////////////////////////
uart_tx inst4_uart_tx
(
 .clk(clk_100MHz),
 .rst(rst),
 .Baud_set(Baud_set),
 .empty(fifo_empty),
 .rd_en(fifo_rd_en),
 .uart_data(uart_data),
 .led_tx(led_tx),
 .uart_tx(uart_tx)
);
endmodule

```

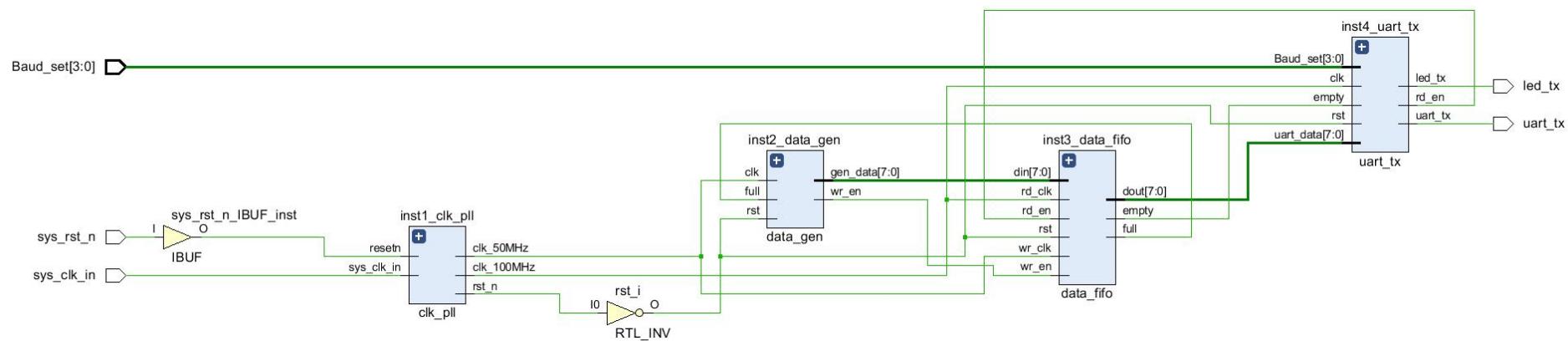
顶层文件  
(Lab9\_UART\_TX\_Top.v)  
完成模块互联

# Lab9串口发送实验



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

顶层文件  
(Lab9\_UART\_TX\_Top.v)  
完成模块互联





## ■数据产生模块

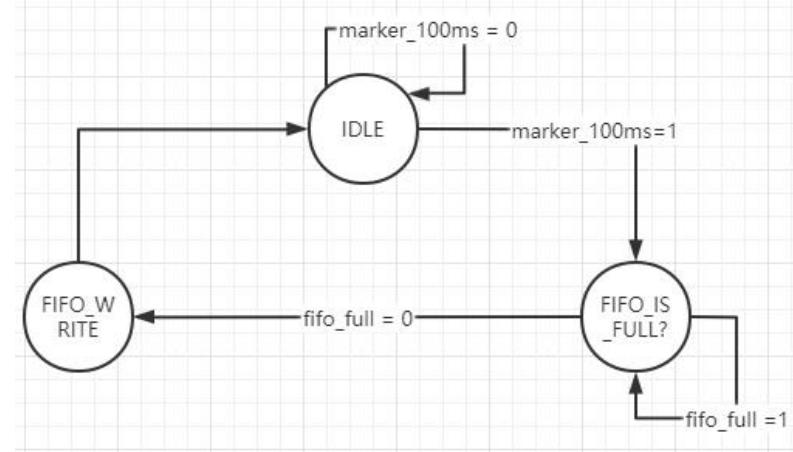
- 设计思想：
  - (1) 设计数据写入节拍 (marker\_100ms) , 每100ms写入一个字节
  - (2) 设计FIFO写入控制状态机, 状态机每循环一次可以向FIFO写入一个字节
  - (3) FIFO满标志 (full) 不是1则可以写入1个字节数据
  - (4) 状态机设计成三个状态: IDLE, FIFO\_IS\_FULL, FIFO\_WRITE
  - (5) 系统复位, 进入IDLE状态
  - (6) 在IDLE状态下, 数据写入节拍有效(marker\_100ms=1), 切换到FIFO\_IS\_FULL状态, 否则状态不变
  - (7) 在FIFO\_IS\_FULL状态下, FIFO满标志无效 (full=0) , 切换到FIFO\_WRITE状态, 否则状态不变
  - (8) 在FIFO\_WRITE状态下, 向FIFO写入一个字节, 次态切换到IDLE状态

# Lab9串口发送实验



## ■ 数据产生模块状态图

- 三个状态: IDLE, FIFO\_IS\_FULL, FIFO\_WRITE
- 状态输出:
  - (1) 复位: fifo写入无效, 写入数据为零
  - (2) IDLE: fifo写入无效, 写入数据为零
  - (3) FIFO\_IS\_FULL: fifo写入信号无效, 写入数据为零
  - (4) FIFO\_WRITE: fifo写入信号有效, 写入数据为学号的一个字节



# Lab9串口发送实验



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

```
module data_gen(
 input clk,
 input rst,
 input full,
 output reg wr_en,
 output reg [7:0] gen_data
);

// 定义状态机的状态
parameter IDLE = 3'b001,
 FIFO_IS_FULL = 3'b010,
 FIFO_WRITE = 3'b100;
reg [3:0] present_state, next_state;

// 生成学号数据 2021EE8014082095
wire [7:0] STU_NUM [0:7];
assign STU_NUM[0] = 8'h20;
assign STU_NUM[1] = 8'h21;
assign STU_NUM[2] = 8'hEE;
assign STU_NUM[3] = 8'h00;
assign STU_NUM[4] = 8'h14; 通过数组预置学号数据
assign STU_NUM[5] = 8'h08;
assign STU_NUM[6] = 8'h20;
assign STU_NUM[7] = 8'h95;

reg [31:0] counter;
// 设置写字节的速度为100ms, clk为50MHz时, 100ms/20ns = 5_000_000
parameter COUNTER_100MS = 32'd5_000_000;
wire marker_100ms;
reg [2:0] i;

// 产生状态机的启动信号, 每10ms启动一次, 同时切换一次数组输出
always @(posedge clk or posedge rst) begin
 if(rst) begin
 counter <= 32'd0;
 end
 else if(counter < (COUNTER_100MS-1)) begin
 counter <= counter + 1'b1;
 end
 else begin
 counter <= 32'd0;
 end
end
assign marker_100ms = (counter == (COUNTER_100MS-1)); 产生数据写入节拍
```

```
// 三段式状态机第一段, 完成现态、次态转换
always @(posedge clk or posedge rst) begin
 if(rst) begin
 present_state <= IDLE;
 end
 else begin
 present_state <= next_state;
 end
end

// 三段式状态机第二段, 完成状态转移判断
always @(*) begin
 case (present_state)
 IDLE : begin
 if(marker_100ms) begin
 next_state <= FIFO_IS_FULL;
 end
 else begin
 next_state <= IDLE;
 end
 end 实现状态机状态转移
 FIFO_IS_FULL: begin
 if(full) begin
 next_state <= FIFO_IS_FULL;
 end
 else begin
 next_state <= FIFO_WRITE;
 end
 end
 FIFO_WRITE: begin
 next_state <= IDLE;
 end
 default: next_state <= IDLE;
 endcase
end
```

```
// 三段式状态机第三段, 完成各状态下动作输出
always @(posedge clk or posedge rst) begin
 if(rst) begin
 wr_en <= 1'b0;
 gen_data <= 8'h00;
 i <= 3'd0;
 end
 else begin
 case(present_state)
 IDLE: begin
 wr_en <= 1'b0;
 gen_data <= 8'h00;
 end
 FIFO_IS_FULL: begin
 wr_en <= 1'b0;
 gen_data <= 8'h00;
 end
 FIFO_WRITE: begin
 wr_en <= 1'b1;
 gen_data <= STU_NUM[i];
 i <= i + 3'd1;
 end
 default: begin
 wr_en <= 1'b0;
 gen_data <= 8'h00;
 end
 endcase
 end
end
```



## ■ 串口发送模块

- 设计思想：

- (1) 设计波特率节拍 (bps\_marker) 保证串口发送时序的波特率
- (2) 设计FIFO读取控制状态机，状态机每循环一次可以从FIFO读出一个字节，将该数据并串转换后通过串口发送
- (3) FIFO空标志为非空 (empty=0) , 则可以读取1个字节数据
- (4) 状态机设计成13个状态: FIFO\_IS\_EMPTY, FIFO\_RD\_EN,FIFO\_READ,  
UART\_TX\_START\_BIT,UART\_TX\_BIT0,UART\_TX\_BIT1,...UART\_TX\_BIT7,UART\_TX\_STOP\_BIT
- (5) 系统复位，进入FIFO\_IS\_EMPTY状态
- (6) 在FIFO\_IS\_EMPTY状态下, FIFO空标志为非空 (empty=0) , 切换到FIFO\_RD\_EN状态，否则状态不变
- (7) 在FIFO\_RD\_EN状态下, FIFO读使能有效，次态切换到FIFO\_READ状态
- (8) 在FIFO\_READ状态下, 读出一个字节，波特率节拍标志有效 (bps\_marker =1), 切换到UART\_TX\_START\_BIT状态
- (9) 在UART\_TX\_START\_BIT状态下, 发送起始位，波特率节拍标志有效 (bps\_marker =1), 切换到UART\_TX\_BIT0状态  
依次发送数据bit...
- (10) 在UART\_TX\_STOP\_BIT状态下, 发送停止位，波特率节拍标志有效 (bps\_marker =1), 切换到FIFO\_IS\_EMPTY状态

# Lab9串口发送实验

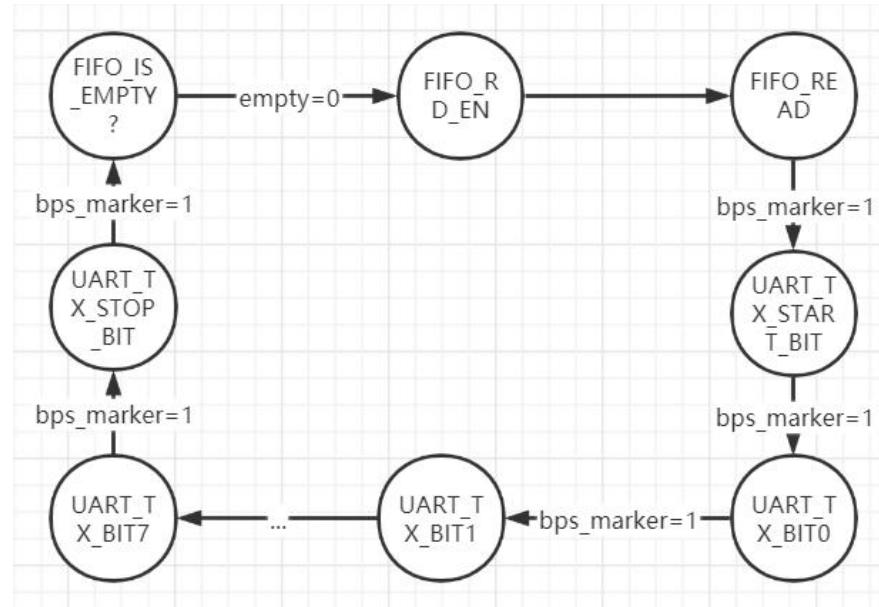


國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study,UCAS

## ■ 串口发送模块状态图

### • 状态输出:

- (1) 复位: fifo读信号无效, 串口 (tx) 输出为高
- (2) FIFO\_IS\_EMPTY: fifo读信号无效, 串口 (tx) 输出为高
- (3) FIFO\_RD\_EN: fifo读信号有效, 串口 (tx) 输出为高
- (4) FIFO\_READ: fifo读信号无效, 从fifo读出一个字节
- (5) UART\_XX: 串口 (tx) 发送相应的bit位



# Lab9串口发送实验



```
module uart_tx(
 input clk,
 input rst,
 input [3:0] Baud_set,
 input empty,
 output reg rd_en,
 input [7:0] uart_data,
 output reg led_tx,
 output reg uart_tx
);

// 定义状态机的状态
parameter FIFO_IS_EMPTY = 16'd0,
 FIFO_RD_EN = 16'd1,
 FIFO_READ = 16'd2,
 UART_TX_START_BIT = 16'd3,
 UART_TX_BIT0 = 16'd4,
 UART_TX_BIT1 = 16'd5,
 UART_TX_BIT2 = 16'd6,
 UART_TX_BIT3 = 16'd7,
 UART_TX_BIT4 = 16'd8,
 UART_TX_BIT5 = 16'd9,
 UART_TX_BIT6 = 16'd10,
 UART_TX_BIT7 = 16'd11,
 UART_TX_STOP_BIT = 16'd12;
reg [4:0] present_state, next_state;

reg [15:0] bps_cnt,bps_cnt_max;
wire bps_marker;
reg [7:0] uart_send_byte; // 信号定义
parameter START_BIT = 1'b0;
parameter STOP_BIT = 1'b1;
```

```
// 产生波特率计数最大值信号，时钟为100MHz时：
// 波特率9600: bps_cnt = (1/9600)s/10ns = 10417
// 波特率19200: bps_cnt = (1/19200)s/10ns = 5208
// 波特率38400: bps_cnt = (1/38400)s/10ns = 2604
// 波特率57600: bps_cnt = (1/57600)s/10ns = 1736
// 波特率115200: bps_cnt = (1/115200)s/10ns = 868
always @(posedge clk or posedge rst) begin
 if(rst) begin
 bps_cnt_max <= 16'd10414;
 end
 else begin
 case (Baud_set)
 4'b0000: bps_cnt_max <= 16'd10414;
 4'b0001: bps_cnt_max <= 16'd5208;
 4'b0010: bps_cnt_max <= 16'd2604;
 4'b0011: bps_cnt_max <= 16'd1736;
 4'b0100: bps_cnt_max <= 16'd868;
 default: bps_cnt_max <= 16'd10414;
 endcase
 end
end
// 产生波特率信号
always @(posedge clk or posedge rst) begin
 if(rst) begin
 bps_cnt <= 16'd0;
 end
 else if(bps_cnt < (bps_cnt_max-1)) begin
 bps_cnt <= bps_cnt + 1'b1;
 end
 else begin
 bps_cnt <= 16'd0;
 end
end
assign bps_marker = (bps_cnt == (bps_cnt_max-1));
```

# Lab9串口发送实验



```
// 三段式状态机第一段，完成现态、次态转换
always @ (posedge clk or posedge rst) begin
 if(rst) begin
 present_state <= FIFO_IS_EMPTY;
 end
 else begin
 present_state <= next_state;
 end
end
//三段式状态机第二段，完成状态转移判断
always @(*) begin
 case (present_state)
 FIFO_IS_EMPTY : begin
 if(empty) begin
 next_state <= FIFO_IS_EMPTY;
 end
 else begin
 next_state <= FIFO_RD_EN;
 end
 end
 FIFO_RD_EN: begin
 next_state <= FIFO_READ;
 end
 FIFO_READ: begin
 if(bps_marker) begin
 next_state <= UART_TX_START_BIT;
 end
 else begin
 next_state <= FIFO_READ;
 end
 end
 UART_TX_START_BIT: begin
 if(bps_marker) begin
 next_state <= UART_TX_BIT0;
 end
 else begin
 next_state <= UART_TX_START_BIT;
 end
 end
 endcase
end
```

```
UART_TX_BIT0: begin
 if(bps_marker) begin
 next_state <= UART_TX_BIT1;
 end
 else begin
 next_state <= UART_TX_BIT0;
 end
end
UART_TX_BIT1: begin
 if(bps_marker) begin
 next_state <= UART_TX_BIT2;
 end
 else begin
 next_state <= UART_TX_BIT1;
 end
end
UART_TX_BIT2: begin
 if(bps_marker) begin
 next_state <= UART_TX_BIT3;
 end
 else begin
 next_state <= UART_TX_BIT2;
 end
end
UART_TX_BIT3: begin
 if(bps_marker) begin
 next_state <= UART_TX_BIT4;
 end
 else begin
 next_state <= UART_TX_BIT3;
 end
end
UART_TX_BIT4: begin
 if(bps_marker) begin
 next_state <= UART_TX_BIT5;
 end
 else begin
 next_state <= UART_TX_BIT4;
 end
end
endcase
```

```
UART_TX_BITS: begin
 if(bps_marker) begin
 next_state <= UART_TX_BIT6;
 end
 else begin
 next_state <= UART_TX_BIT5;
 end
end
UART_TX_BIT6: begin
 if(bps_marker) begin
 next_state <= UART_TX_BIT7;
 end
 else begin
 next_state <= UART_TX_BIT6;
 end
end
UART_TX_BIT7: begin
 if(bps_marker) begin
 next_state <= UART_TX_STOP_BIT;
 end
 else begin
 next_state <= UART_TX_BIT7;
 end
end
UART_TX_STOP_BIT: begin
 if(bps_marker) begin
 next_state <= FIFO_IS_EMPTY;
 end
 else begin
 next_state <= UART_TX_STOP_BIT;
 end
end
default: next_state <= FIFO_IS_EMPTY;
```

# Lab9串口发送实验



```
//三段式状态机第三段，完成各状态下动作输出
always @(posedge clk or posedge rst) begin
 if(rst) begin
 rd_en <= 1'b0;
 uart_tx <= 1'b1;
 uart_send_byte <= 8'hFF;
 led_tx <= 1'b0;
 end
 else begin
 case(present_state)
 FIFO_IS_EMPTY: begin
 rd_en <= 1'b0;
 uart_tx <= 1'b1;
 led_tx <= 1'b0;
 end
 FIFO_RD_EN: begin
 rd_en <= 1'b1;
 uart_tx <= 1'b1;
 led_tx <= 1'b0;
 end
 FIFO_READ: begin
 rd_en <= 1'b0;
 uart_tx <= 1'b1;
 uart_send_byte <= uart_data;
 led_tx <= 1'b0;
 end
 end
 UART_TX_START_BIT: begin
 rd_en <= 1'b0;
 uart_tx <= START_BIT;
 led_tx <= 1'b1;
 end
 UART_TX_BIT0: begin
 rd_en <= 1'b0;
 uart_tx <= uart_send_byte[0];
 led_tx <= 1'b1;
 end
 UART_TX_BIT1: begin
 rd_en <= 1'b0;
 uart_tx <= uart_send_byte[1];
 led_tx <= 1'b1;
 end
 endcase
 end
end
```

```
UART_TX_BIT2: begin
 rd_en <= 1'b0;
 uart_tx <= uart_send_byte[2];
 led_tx <= 1'b1;
end
UART_TX_BIT3: begin
 rd_en <= 1'b0;
 uart_tx <= uart_send_byte[3];
 led_tx <= 1'b1;
end
UART_TX_BIT4: begin
 rd_en <= 1'b0;
 uart_tx <= uart_send_byte[4];
 led_tx <= 1'b1;
end
UART_TX_BIT5: begin
 rd_en <= 1'b0;
 uart_tx <= uart_send_byte[5];
 led_tx <= 1'b1;
end
UART_TX_BIT6: begin
 rd_en <= 1'b0;
 uart_tx <= uart_send_byte[6];
 led_tx <= 1'b1;
end
UART_TX_BIT7: begin
 rd_en <= 1'b0;
 uart_tx <= uart_send_byte[7];
 led_tx <= 1'b1;
end
UART_TX_STOP_BIT: begin
 rd_en <= 1'b0;
 uart_tx <= STOP_BIT;
 led_tx <= 1'b1;
end
default: begin
 rd_en <= 1'b0;
 uart_tx <= 1'b1;
 led_tx <= 1'b0;
end
endcase
end
endmodule
```

# 下节预告



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

## Lab10 实验内容

# Lab10串口接收实验



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

## ■实验内容

- 通过上位机发送指令控制led灯的亮灭
- 具体指令及功能如右图所示
- 串口接收模块接收串口数据，将有效数据存入到FIFO中，led控制模块根据FIFO状态读出数据后进行解析，执行相应的动作
- 附加功能：动作执行完毕后，将执行状态通过串口反馈回上位机

## ■实验目的

- 进一步熟悉状态机的使用
- 建立模块化设计思维方式
- 理解串口接收时序逻辑

| 序号 | 指令代码 | 执行动作    | 执行状态 |
|----|------|---------|------|
| 1  | 0x01 | LD1_1熄灭 | 0xE1 |
| 2  | 0x11 | LD1_1点亮 | 0xF1 |
| 3  | 0x02 | LD1_2熄灭 | 0xE2 |
| 4  | 0x12 | LD1_2点亮 | 0xF2 |
| 5  | 0x03 | LD1_3熄灭 | 0xE3 |
| 6  | 0x13 | LD1_3点亮 | 0xF3 |
| 7  | 0x04 | LD1_4熄灭 | 0xE4 |
| 8  | 0x14 | LD1_4点亮 | 0xF4 |



## 自行设计：

1. 根据实验要求形成设计思路
2. 完成电路框图设计（模块划分、模块功能、模块接口）
3. 整理输入输出接口表
4. 完成设计输入（HDL、设计约束、仿真文件）
5. 核对RLT电路图、完成功能仿真
6. 完成调试（嵌入式逻辑分析仪、实际效果验证）



# THANKS



國科大杭州高等研究院  
Hangzhou Institute for Advanced Study, UCAS



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS



# FPGA电路软硬件设计 第十二讲 (12.1 FPGA项目开发及管理)

2025年5月20日

# 本节内容



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study,UCAS

**以面向载人航天FPGA开发为例  
介绍FPGA项目开发及管理**



- 1. 载人航天FPGA软件开发规范**
- 2. 载人航天FPGA软件需求分析**
- 3. 载人航天FPGA软件安全编程**
- 4. 载人航天FPGA软件测试验证**

# 本节内容



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study,UCAS

## 1. 载人航天FPGA软件开发规范

# 1.载人航天FPGA软件开发规范



## ■ 载人航天FPGA软件开发的特殊性

### • 特殊性一：载人航天造价昂贵

- ① “天舟系列”货运飞船，单次的成本价格约为3.5亿元人民币
- ② “神舟系列”载人飞船，单次的成本价格约为8.5亿元人民币
- ③ “长征3号”运载火箭，单次发射费用约为7000万美元

### • 特殊性二：国之重器，不容有失

- ① 中国空间站已在轨实施了130多项科学研究与应用项目
- ② 在空间生命科学、航天医学、空间材料科学、微重力流体物理等方向已取得重要成果，在国际一流期刊发表论文280余篇

### • 特殊性三：太空环境恶劣，对宇航产品质量要求高

- ① 低温高温：温度能低至-100°以下，温度可高达100°以上
- ② 强辐射：太阳宇宙线辐射、高速粒子，单粒子翻转效应
- ③ 高真空：太空中每立方厘米只有0.1个氢原子



向天时部相机1



向全景摄像机b



# 1.载人航天FPGA软件开发规范

- **FPGA**——在载人航天项目中称之为可编程逻辑产品，固化的代码——称之为配置项
- **FPGA项目开发划归为软件范畴，研制过程需要符合建造规范要求且严格受控**



# 1.载人航天FPGA软件开发规范



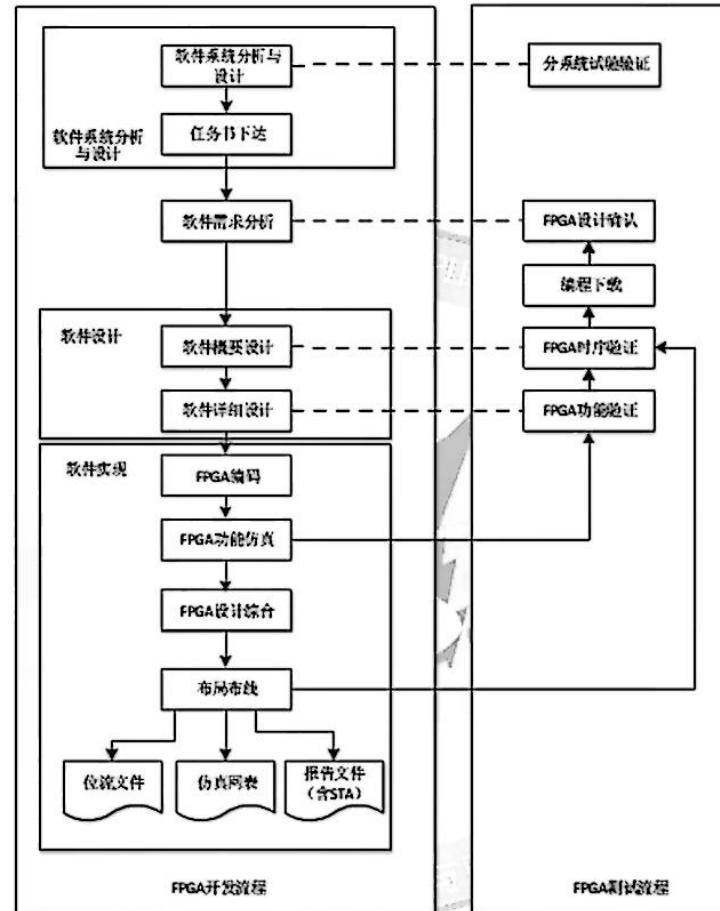
## ■ FPGA项目开发的基本流程

### • FPGA开发流程：

软件系统分析与设计、软件需求分析、软件设计、软件实现

### • FPGA测试流程：

FPGA功能验证，FPGA时序验证，编程下载，FPGA设计确认，分系统试验验证



# 1.载人航天FPGA软件开发规范



完成关键技术攻关及验证  
完成系统级原理验证  
相当于完成原理样机

分为初样电性件及初样鉴定件  
电性件实现技术见底  
鉴定件与正样件状态一致  
完成所有鉴定级试验  
相当于完成原型样机

发射件产品  
全过程可追溯，质量优先

# 1.载人航天FPGA软件开发规范



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study,UCAS

■ 通常按照系统层、项目层、研制组来分级管理

方案阶段

| 序号 | 软件工程阶段 | 任务主体 | 软件研制过程   | 工作要求                                                                                                                                          | 阶段输入               | 阶段输出                                                                              | 完成标志                                                   |
|----|--------|------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------|--------------------|-----------------------------------------------------------------------------------|--------------------------------------------------------|
| F1 | 方案阶段   | 系统层  | 系统分析与设计  | <p>①进行信息系统任务分析和分解、信息流分析、架构设计、工作模式设计、任务剖面设计、信息接口设计等工作；</p> <p>②依据信息接口设计结果确定各分系统之间的数据流、指令流和接口信息协议；</p> <p>③进行系统初步危险分析和安全性分析，明确分系统可靠性、安全性要求。</p> | -                  | <p>《系统方案设计报告》（含软件系统设计方案、系统初步危险分析）</p> <p>《系统方案研制报告》</p> <p>《分系统研制任务书》（初步）</p>     | <p>①文档齐全；</p> <p>②通过系统方案设计评审；</p> <p>③纳入系统技术状态控制。</p>  |
| F2 | 方案阶段   | 项目层  | 分系统分析与设计 | <p>①初步分析软件系统外部信息流，确定软件系统主要功能、性能指标和工作模式；</p> <p>②初步设计软件系统体系结构等。</p>                                                                            | <p>《系统方案设计报告》</p>  | <p>《分系统方案设计报告》（含初步软件系统设计、软件系统初步危险分析）</p>                                          | <p>①文档齐全；</p> <p>②通过分系统方案设计评审；</p> <p>③纳入系统技术状态控制。</p> |
| F3 | 方案阶段   | 研制组  | 原型软件研制   | 配合关键技术攻关，研制核心原型软件。                                                                                                                            | <p>《分系统方案设计报告》</p> | <p>《原型软件需求规格说明》（视需要）</p> <p>《原型软件代码》</p> <p>《原型软件测试报告》（视需要）</p> <p>《原型软件版本说明》</p> | <p>①文档齐全；</p> <p>②通过分系统主任设计师或主管软件副主任设计师批准。</p>         |



# 1.载人航天FPGA软件开发规范

| 序号 | 软件工程阶段 | 任务主体       | 软件研制过程       | 工作要求                                                                                                                                                                                               | 阶段输入                                                             | 阶段输出                                                                                                      | 完成标志                                                            |
|----|--------|------------|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|
| C1 | 初样阶段   | 系统层        | 系统分析与设计      | 对方案阶段的系统分析与设计进行修改完善。                                                                                                                                                                               | 《系统方案设计报告》<br>《系统方案研制报告》                                         | 《系统初样设计报告》(含软件系统设计方案)<br>《分系统研制任务书》                                                                       | ①文档齐全；<br>②通过系统初样设计评审；<br>③纳入系统技术状态控制。                          |
| C2 | 初样阶段   | 项目层        | 分系统分析与设计     | ①软、硬件功能分配，分析软件系统外部信息流，确定软件系统主要功能、性能指标和工作模式；<br>②设计软件系统体系结构，确定软件配置项间数据流、指令流、时序关系和接口信息协议；<br>③合理划分软件配置项，选择适应的处理器或可编程逻辑器件，并进行必要性、设计需求分析；分析、确定软件配置项等级；<br>④在系统初步危险分析的基础上，进行分系统危险分析，并提出软件配置项的安全性、可靠性要求。 | 《分系统研制任务书》<br>《分系统方案设计报告》<br>《分系统方案研制报告》                         | 《软件系统设计说明》<br>《软件系统危险分析报告》或《分系统危险分析报告》<br>《软件系统测试大纲》或《分系统测试大纲》<br>《软件接口需求与设计说明》<br>分系统《软件产品配套表》、《软件等级申请表》 | ①文档齐全；<br>②通过分系统分析与设计评审。<br>③输出产品按照要求纳入软件受控库；<br>④软件研制任务书下达研制组。 |
| C3 | 初样阶段   | 项目层<br>系统层 | 软件研制任务书编制与下达 | ①明确软件配置项外部接口、功能、性能、软件工作模式、设计约束及安全可靠性等要求；<br>②明确编程语言、编译器、操作系统等软件开发、运行环境要求及测试要求，明确研制应遵循的标准和规范；<br>③明确软件等级、开发进度、质量保证等管理要求，并提出验收交付及维护要求。                                                               | 《软件系统设计说明》<br>《软件系统危险分析报告》或《分系统危险分析报告》<br>《软件接口需求与设计说明》<br>(视需要) | 《软件研制任务书》<br>《软件第三方评测任务书》(针对A、B级软件)                                                                       |                                                                 |

初样阶段



# 1.载人航天FPGA软件开发规范

初样阶段

| 序号 | 软件工程阶段 | 任务主体 | 软件研制过程             | 工作要求                                                                                                                              | 阶段输入                                                    | 阶段输出                                                                                             | 完成标志                                    |
|----|--------|------|--------------------|-----------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------|--------------------------------------------------------------------------------------------------|-----------------------------------------|
| C4 | 初样阶段   | 研制组  | 软件需求分析             | 根据软件研制任务书，确定被开发软件的功能、性能、接口、可靠性、安全性、以及运行环境等要求。                                                                                     | 《软件研制任务书》<br>《软件接口需求与设计说明》<br>《基于软件功能的安全性分析报告》<br>(视需要) | 《软件需求规格说明》<br>《软件接口需求与设计说明》<br>《基于软件功能的安全性分析报告》<br>《软件配置项测试计划》<br>《软件开发计划》×《软件配置管理计划》×《软件质量保证计划》 | ①文档齐全；<br>②通过软件需求分析评审；<br>③按照要求纳入软件受控库。 |
| C5 | 初样阶段   | 研制组  | 软件设计               | 处理器软件：<br>①根据软件需求规格说明，设计软件总体结构、处理流程、运行状态；<br>②划分并定义软件部件，以及各部件的数据接口、控制接口，设计全局数据库和数据结构。                                             | 《软件需求规格说明》                                              | 《软件概要设计说明》<br>《基于软件部件的安全性分析报告》<br>《数据库设计说明》(视需要)<br>《软件集成测试计划》                                   | ①文档齐全；<br>②通过软件概要设计评审；<br>③按照要求纳入软件受控库。 |
|    |        |      |                    | 可编程逻辑：<br>①进行时钟设计、复位设计、跨时钟域同步设计、状态机设计、流水线设计等。<br>进行层次结构设计；<br>②确定部件接口关系，包括信号、连接关系、时序关系、协议等定义；<br>③进行部件设计，包括模块区域设计，部件（或模块）区域设计规划等。 |                                                         |                                                                                                  |                                         |
| C6 | 初样阶段   | 研制组  | 软件详细设计             | 对软件概要设计中产生的部件进行细化设计，划分并定义软件单元，设计单元的内部细节，包括程序模型算法和数据结构，为编写源代码提供必要的说明。                                                              | 《软件概要设计说明》<br>《数据库设计说明》(视需要)<br>《基于软件部件的安全性分析报告》        | 《软件详细设计说明》<br>《基于软件单元的安全性分析报告》<br>《软件单元测试计划》                                                     | ①文档齐全；<br>②通过软件详细设计评审；<br>③按照要求纳入软件受控库。 |
| C7 | 初样阶段   | 研制组  | 软件实现 <sup>±3</sup> | 处理器软件：<br>根据软件详细设计说明，进行软件编程、调试，开展静态分析、代码审查、                                                                                       | 《软件详细设计说明》<br>《基于软件单元的安全性分析报告》                          | 软件源代码<br>软件可执行程序<br>《软件单元测试说明》(含辅助程序)                                                            | ①文档齐全；<br>②通过软件实现及单元测试评审；               |



# 1.载人航天FPGA软件开发规范

初样阶段

| 序号 | 软件工程阶段 | 任务主体 | 软件研制过程 | 工作要求                                                                                                                                                                     | 阶段输入                                   | 阶段输出                                                                                                                         | 完成标志                                     |
|----|--------|------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------|------------------------------------------|
|    |        |      |        | 代码走查和单元测试，验证软件单元与设计说明的一致性。<br><br>可编程逻辑：<br>根据软件详细设计说明，进行编码、功能验证、设计综合、布局布线，并开展静态分析、代码审查、代码走查等静态测试。                                                                       | 《软件单元测试计划》                             | 《软件单元测试报告》<br>《软件静态测试报告》<br>《软件中断分析报告》                                                                                       | ③按照要求纳入软件受控库。                            |
| C8 | 初样阶段   | 研制组  | 软件测试   | 处理器软件：<br>按照软件概要设计说明和详细设计说明中规定的软件结构，将软件单元逐步集成成为软件部件直至软件配置项，重点检查软件单元之间和软件部件之间的接口和工作的协调性。<br><br>可编程逻辑：<br>按照软件概要设计与详细设计说明进行静态时序分析、动态时序仿真、等效性检查等，重点检查可编程逻辑门级功能、性能、时序等的正确性。 | 通过单元测试软件代码<br>《软件单元测试报告》<br>《软件集成测试计划》 | 《软件集成测试说明》(含辅助程序)<br>《软件集成测试报告》<br>《软件静态测试报告》                                                                                | ①文档齐全；<br>②通过软件集成测试评审；<br>③按照要求纳入软件受控库。  |
| C9 | 初样研制阶段 | 研制组  |        | 应用使用与开发相同的编译环境，被测软件编程下载，在真实目标机上或真实环境中运行，外围可以是实物、半实物和仿真接口，确认该软件是否达到了软件需求规格说明所规定的各项要求。                                                                                     |                                        | 《软件配置项测试说明》(含辅助程序)<br>《软件配置项测试报告》<br>《软件静态测试报告》<br>《软件固化落焊记录单》<br>《软件固化落焊测试报告》<br>《软件版本说明》<br>《软件操作手册》(视需要)<br>《软件用户手册》(视需要) | ①文档齐全；<br>②通过软件配置项测试评审；<br>③按照要求纳入软件受控库。 |



# 1.载人航天FPGA软件开发规范

初样阶段

| 序号  | 软件工程阶段 | 任务主体       | 软件研制过程  | 工作要求                                                                                                                       | 阶段输入                                                                                                                                          | 阶段输出                                         | 完成标志                                                            |
|-----|--------|------------|---------|----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------|-----------------------------------------------------------------|
|     |        |            |         |                                                                                                                            | 《软件程序员手册》(视需要)<br>《固件保障手册》(视需要)                                                                                                               |                                              |                                                                 |
| C10 | 初样阶段   | 项目层        | 分系统试验验证 | 将各个软件配置项集成，从任务角度对软件系统进行测试，重点测试软件按照任务剖面运行时的状态、软件配置项间的配合以及软硬件之间协同的情况。                                                        | 通过配置项测试的软件代码<br>《软件系统测试大纲》或《分系统测试大纲》                                                                                                          | 《软件系统测试说明》或《分系统测试说明》<br>《软件系统测试报告》或《分系统测试报告》 | ①文档齐全；<br>②通过软件系统测试相关评审；<br>③按照要求纳入分系统软件受控库；产品基线形成后，纳入分系统软件产品库。 |
| C11 | 初样阶段   | 项目层        | 软件研制总结  | 对软件研制、软件重大技术状态变更、软件质量复查、软件工程化管理等方面进行总结，A、B级软件须形成独立的总结报告。                                                                   | 软件研制阶段输出产品                                                                                                                                    | 《软件研制总结报告》                                   | ①文档齐全；<br>②通过所评审。                                               |
| C12 | 初样阶段   | 项目层<br>系统层 | 验收交付    | ①项目(或承研单位)应根据软件研制任务书的要求对程序、文档、数据等进行整理，并编制相应的总结报告；<br>②交办方根据软件研制任务书对软件研制过程中形成的程序、文件、测试结果等进行检查确认，必要时开展验收测试，并按照规定履行正式的验收交付手续。 | 《软件产品配套表》<br>《软件研制任务书》<br>《软件产品清单》 <sup>注1</sup><br>各阶段软件产品<br>《软件研制总结报告》 <sup>注1</sup><br>《软件产品证明书》 <sup>注1</sup><br>《软件产品履历书》 <sup>注2</sup> | 《验收报告》(含软件验收内容)                              | ①验收交付完成；<br>②软件产品纳入系统软件受控库。                                     |
| C13 | 初样阶段   | 系统层        | 系统试验验证  | 系统试验验证包括空间应用系统联试、整船或整器综合测试、跨系统接口对接、大系统联调等大型试验。                                                                             | 《空间应用系统各种大型试验大纲》<br>系统软件受控库提取的软件产品                                                                                                            | 《空间应用系统各种大型试验测试细则》<br>《空间应用系统各种大型试验总结报告》     | ①文档齐全；<br>②通过系统测试相关评审；<br>③软件产品纳入系统软件产品库。                       |

# 1.载人航天FPGA软件开发规范



初样阶段

| 序号  | 软件工程阶段 | 任务主体              | 软件研制过程 | 工作要求                                                             | 阶段输入       | 阶段输出          | 完成标志                        |
|-----|--------|-------------------|--------|------------------------------------------------------------------|------------|---------------|-----------------------------|
| C14 | 初样阶段   | 系统层               | 软件研制总结 | 对软件总体设计、软件研制、软件重大技术状态变更、软件质量复查、软件工程化管理等方面进行总结，A、B 级软件须形成独立的总结报告。 | 软件研制阶段输出产品 | 《软件研制总结报告》    | ①文档齐全；<br>②通过出厂评审。          |
| C15 | 初样阶段   | 系统层<br>项目层<br>研制组 | 运行维护   | 软件产品交付后，需要进行改正性维护、改善性、新增型的维护，运行维护要严格受控。                          | 软件研制阶段输出产品 | 变更后的各研制阶段输出产品 | ①重大变化通过相应的评审；<br>②纳入软件配置管理。 |



# 1.载人航天FPGA软件开发规范

正样阶段

| 序号     | 软件工程阶段 | 任务主体              | 软件研制过程                         | 工作要求                                                             | 阶段输入                                                         | 阶段输出                                                                                                                     | 完成标志                                                          |
|--------|--------|-------------------|--------------------------------|------------------------------------------------------------------|--------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|
| Z1     | 正样阶段   | 系统层               | 系统分析与设计                        | 对初样阶段的系统分析与设计进行修改完善。                                             | 《系统初样设计报告》<br>《系统初样研制报告》                                     | 《系统正样设计报告》(含软件系统设计方案)<br>《分系统研制任务书》                                                                                      | ①文档齐全；<br>②通过系统设计评审；<br>③纳入系统技术状态控制。                          |
| Z2     | 正样阶段   | 项目层               | 分系统分析与设计                       | 对初样阶段的软件系统分析与设计进行修改完善。                                           | 《分系统研制任务书》<br>《分系统初样设计报告》<br>《分系统初样研制报告》                     | 《软件系统设计说明》(修订)<br>《软件系统危险分析报告》或《分系统危险分析报告》(修订)<br>《软件系统测试大纲》或《分系统测试大纲》(修订)<br>《软件接口需求与设计说明》<br>分系统《软件产品配套表》《软件等级申请表》(修订) | ①文档齐全；<br>②通过分系统设计与分析评审(按照变更类别组织相应评审)。<br>③按照要求纳入软件受控库，进行变更入库 |
| Z3     |        | 项目层<br>系统层        | 软件研制任务书编制与下达                   | 对初样阶段的软件研制任务书修改完善；<br>有修订则重新下达任务书，无修订则无需重新下达任务书。                 | 《软件系统设计说明》<br>《软件系统危险分析报告》或《分系统危险分析报告》<br>《软件接口需求与设计说明》(视需要) | 软件研制任务书(修订)                                                                                                              | ①通过分系统设计与分析评审(按照变更类别组织相应评审)。<br>②修订任务书下达。                     |
| Z4~Z15 | 正研阶段   | 研制组               | 软件需求分析<br>软件设计<br>软件实现<br>软件测试 | 变更或增量开发相应的软件功能，包括对初样版本的修改，也包括对新增功能的研制，直到产生最终的产品，并开展全面的软件第三方评测工作。 | 同初样                                                          | 同初样                                                                                                                      | ①文档齐全；<br>②通过相应的评审；<br>③按照要求纳入软件受控库。                          |
| Y1~Y15 | 在轨运行阶段 | 系统层<br>项目层<br>研制组 | 在轨运行维护                         | 软件产品交付后，需要进行改正性维护、改善性、新增型的维护，运行维护要严格受控。                          | 软件研制阶段输出产品                                                   | 变更后的各研制阶段输出产品                                                                                                            | ①重大变化通过相应的评审；<br>②纳入软件配置管理。                                   |

# 1.载人航天FPGA软件开发规范



# 國科大杭州高等研究院 Hangzhou Institute for Advanced Study, UCAS

#### ■ 各类文件严格受控

- 开发库, 受控库, 产品库
  - 研制组, 项目层, 系统层

| 软件研发阶段   | 产生的软件配置管理项             | 研制组 | 项目(或研制单位)层 |     | 系统层 | 受控库基线 |   |   |   |   |   |
|----------|------------------------|-----|------------|-----|-----|-------|---|---|---|---|---|
|          |                        |     | 开发库        | 受控库 |     |       | △ | △ | - | - | - |
| 分系统分析与设计 | 软件系统设计说明               | -   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | - | - |
|          | 软件接口需求说明               | -   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 软件接口设计说明               | -   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 软件系统危险分析报告(或分系统危险分析报告) | -   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 软件系统测试大纲(分系统测试大纲)      | -   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 软件研制任务书                | -   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 评审相关证明材料               | -   | △          | -   | -   | △     | - | △ | - | △ | - |
|          | 软件配置管理相关记录             | -   | △          | -   | -   | △     | - | △ | - | △ | - |
| 软件需求分析   | 软件开发计划                 | ✓   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 软件质量保证计划               | ✓   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 软件配置管理计划(可合并到开发计划)     | ✓   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 软件需求规格说明(含数据库需求)       | ✓   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 基于软件功能安全性分析报告          | ✓   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 软件配置项测试计划              | ✓   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 评审相关证明材料               | -   | △          | -   | -   | △     | - | △ | - | △ | - |
|          | 软件配置管理相关记录             | -   | △          | -   | -   | △     | - | △ | - | △ | - |
| 软件概要设计   | 软件概要设计说明(含数据库设计)       | ✓   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 基于软件部件安全性分析报告          | ✓   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 软件部件测试计划               | ✓   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 评审相关证明材料               | -   | △          | -   | -   | △     | - | △ | - | △ | - |
|          | 软件配置管理相关记录             | -   | △          | -   | -   | △     | - | △ | - | △ | - |
| 软件详细设计   | 软件详细设计说明               | ✓   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 基于软件单元安全性分析报告          | ✓   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 软件单元测试计划               | ✓   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 评审相关证明材料               | -   | △          | -   | -   | △     | - | △ | - | △ | - |
|          | 软件配置管理相关记录             | -   | △          | -   | -   | △     | - | △ | - | △ | - |
| 软件实现     | 软件源代码                  | ✓   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 软件可执行程序                | ✓   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 软件运行所需数据及支持软件(视需要)     | ●   | ●          | ●   | ●   | ●     | △ | △ | - | △ | - |
|          | 软件开发、调试的软件成分及配置说明      | △   | △          | -   | -   | -     | △ | △ | - | △ | - |
|          | 软件开发过程日志               | △   | △          | -   | -   | -     | △ | △ | - | △ | - |
|          | 软件单元测试说明               | ✓   | ✓          | ✓   | ✓   | ✓     | △ | △ | - | △ | - |
|          | 软件单元测试相关的数据、辅助软件、配置说明等 | △   | △          | -   | -   | -     | △ | △ | - | △ | - |
|          | 各种接口规范(视需要)            | △   | △          | -   | -   | -     | △ | △ | - | △ | - |
| 软件实现     | 软件产品交付清单               | -   | -          | -   | -   | -     | △ | △ | - | △ | - |
|          | 软件研制总结报告               | -   | -          | -   | -   | -     | △ | △ | - | △ | - |
|          | 软件产品证明书                | -   | -          | -   | -   | -     | △ | △ | - | △ | - |
|          | 软件产品履历书                | -   | -          | -   | -   | -     | △ | △ | - | △ | - |
|          | 验收报告(含软件验收内容)          | -   | -          | -   | -   | -     | △ | △ | - | △ | - |
|          | 各种接口规范(视需要)            | △   | △          | -   | -   | -     | △ | △ | - | △ | - |
|          | 软件工程标准(视需要)            | △   | △          | -   | -   | -     | △ | △ | - | △ | - |
|          | 其他                     | -   | -          | -   | -   | -     | △ | △ | - | △ | - |

# 本节内容



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study,UCAS

## 2. 载人航天FPGA软件需求分析

## 2. 载人航天FPGA软件需求分析



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

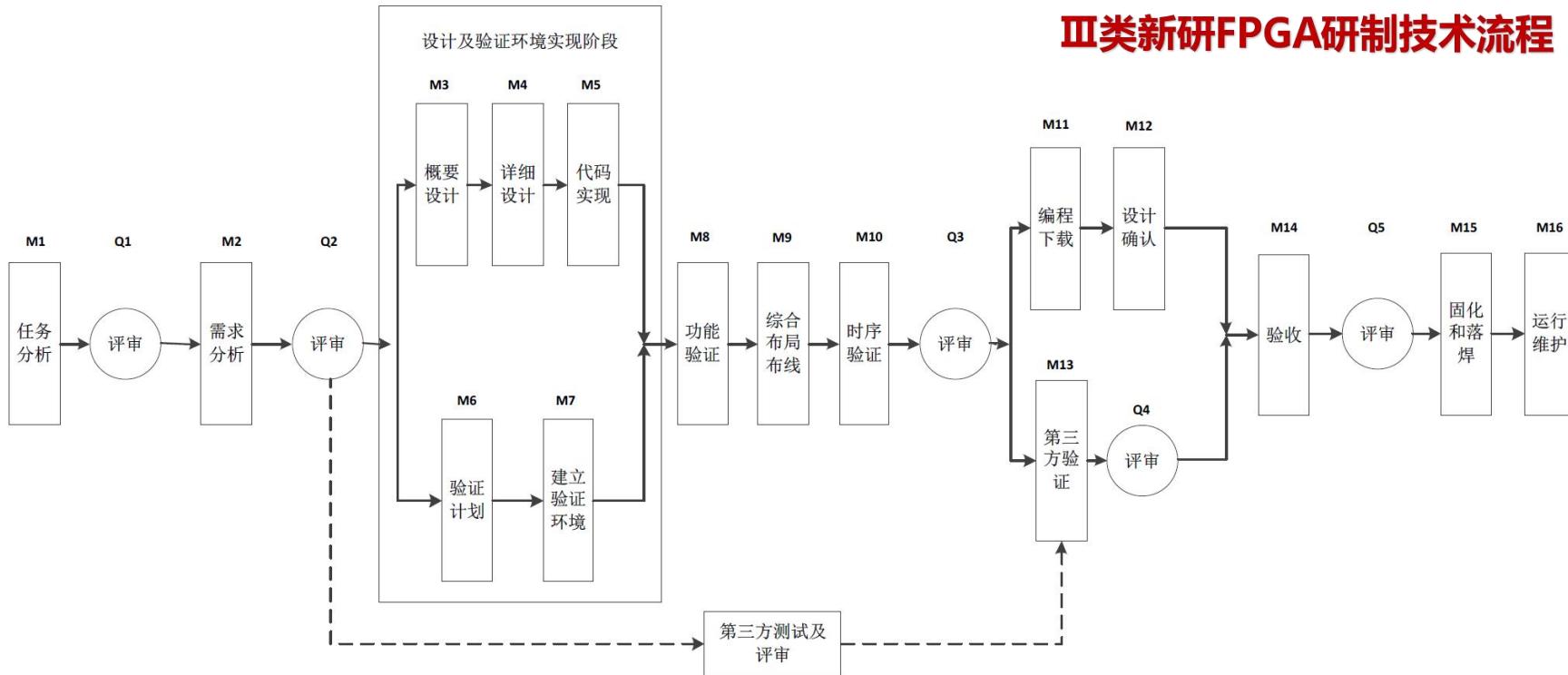
### ■ FPGA按照其成熟情况分为三类

- **I类**: 沿用FPGA(不加修改即可再次使用的FPGA)
- **II类**: 继承FPGA, 修改所涉及的行数占比不超过30%
- **III类**: 不满足 I 类、 II类条件的新研FPGA

## 2. 载人航天FPGA软件需求分析



### III类新研FPGA研制技术流程



流程中包括5个质量控制点 (Q1~Q5) , 每个质量控制点都需要正式评审

## 2. 载人航天FPGA软件需求分析



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

### ■ 主要工作：

- 根据任务书开展FPGA软件需求分析；
- 制定开发计划、配置管理计划；
- 编写需求规格说明。

### ■ 技术要求：

- 分析FPGA实现功能、性能和代码的继承性，确定研制技术流程；
- 确认涉及到的设计验证人员、分工和接口；
- 列出阶段时间安排和所需提交的阶段成果；
- 确认FPGA设计、仿真、综合、布局布线工具以及工具之间的接口；
- 确定FPGA设计输入方式和IP核资源；

## 2. 载人航天FPGA软件需求分析



### ■ 技术要求：

- 确定仿真验证方式和可复用的测试平台、仿真模型等资源；
- 确认所选用的FPGA满足质量等级、抗辐射等级、工作频率、封装等要求；
- 确定FPGA逻辑资源占用率要求，即宇航应用的FPGA等效逻辑门使用率不超过 80%，地面应用的FPGA可参照执行；
- 确定FPGA时钟余量要求，即宇航应用的FPGA至少保持 20% 的时钟余量，地面应用的FPGA 可参照执行。时钟余量的计算方法为  $(\text{max-typ}) / \text{max} * 100\%$ ，其中max为理论最大工作时钟频率，typ为实际运行工作时钟频率；
- 细化研制任务书的相关内容，**FPGA设计人员应能够根据需求规格说明完成设计工作**；

## 2. 载人航天FPGA软件需求分析



### ■ 技术要求：

- 进行需求跟踪，对所有需求项进行标识，包括功能、时序、可靠性安全性、性能需求等，便于进行追踪和覆盖性检查；
- 如果采用的SRAM类型FPGA需要进行配置刷新或三模冗余设计，应按相关技术要求制定计划在各研制阶段开展；
- 如果采用了 SRAM 类型 FPGA，应按相关技术要求制定计划移除半锁存器（Half\_Latch）。
- 注：SRAM型FPGA进行单粒子效应防护时，针对特殊的电路结构，如针对VirtexII系列器件的 Half\_Latch，由于用户不可访问该类资源，所以需要移除半锁存器（Half\_Latch），在硬件电路设计时从FPGA 引出两个 IO 分别上拉和下拉。

## 2. 载人航天FPGA软件需求分析



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

### ■ 主要的阶段产品

- FPGA软件开发计划
- FPGA软件需求规格说明
- FPGA配置管理计划及质量保证计划

## 2. 载人航天FPGA软件需求分析



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

- (1) FPGA软件开发计划内容及编写要求
- (2) FPGA软件需求规格说明内容及编写要求
- (3) FPGA配置管理计划及质量保证计划

## 2. 载人航天FPGA软件需求分析



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

### (1) FPGA软件开发计划内容及编写要求

## 2. 载人航天FPGA软件需求分析



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study,UCAS

### ■ 前沿

- 简述任务来源、FPGA应用环境、FPGA代号等内容

### ■ 引用文档

- 需求规格说明引用的编制规范约束性文件的名称和编号
- 相比于需求规格说明，**无需引用技术性研制依据文件**

### ■ 符号和缩略语

- 列出本文档中用到的专门术语和缩略语定义

### ■ 组织和职责

- 确认参加项目开发的设计验证人员、分工和接口

## 2. 载人航天FPGA软件需求分析



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

### ■ 开发计划

- 参照研制任务书和921《FPGA软件研制技术要求》的要求，列出阶段时间安排、工作内容和所需提交的阶段结果

### ■ 资源保障

- 确认FPGA设计、仿真、综合、布局布线工具以及工具之间的接口
- 确定IP核资源、可复用的测试平台、仿真模型等资源

### ■ 风险分析

- 确认所选用的FPGA满足质量等级、抗辐照等级、设计门数、工作频率、封装等要求和可能存在的采购风险
- 列举可能存在的其他风险

## 2. 载人航天FPGA软件需求分析



### ■ 对照开发计划检查表查看任务是否明确

1、设计、验证人员是否明确？

组织、职责

2、分工是否明确？

3、人员之间的工作接口是否明确？

1、阶段安排是否合理？

开发过程

2、工作内容是否明确？

3、各阶段所需提交结果是否明确？

1、FPGA设计、仿真、综合、布局布线工具是否明确？

资源保障

2、需购买的IP核是否到位？

3、是否有现成可用的测试激励、仿真模型？

1、所选用的FPGA是否满足质量等级、抗辐照等级、设计门数、工作频率、封装等是否已经确认？

风险分析

2、是否存在采购风险？

## 2. 载人航天FPGA软件需求分析



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

### (2) FPGA软件需求规格说明内容及编写要求

## 2. 载人航天FPGA软件需求分析



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

### ■ 前沿

- 简述任务来源、FPGA应用环境、FPGA代号等内容

### ■ 引用文档

- 需求规格说明引用的编制规范约束性文件的名称和编号
- 应包含**研制任务书、FPGA软件研制技术要求**（载人航天）、**FPGA产品保证技术要素等**，引用的技术性研制依据文件的名称和编号，需明确文件的**版本号**

### ■ 符号和缩略语

- 列出本文档中用到的专门术语和缩略语定义

## 2. 载人航天FPGA软件需求分析



### ■ FPGA概述

- 简述FPGA运行环境、FPGA研制技术流程、FPGA功能概述、开发环境概述

#### FPGA运行环境

FPGA芯片型号、程序存储器型号、FPGA加载方式、工作温度、范围要求等。

#### FPGA研制技术流程

FPGA安全关键级别、FPGA研制类型及相应的设计流程控制图，表明关键质量控制点

#### FPGA功能概述

FPGA产品系统组成，可用方框图标示接口示意及内部结构，描述FPGA产品对于外部硬件输入输出接口关系图，包括数据流和控制流。  
对FPGA产品各外围器件、设备及协议进行描述。

#### 开发环境概述

根据需要可规定使用的FPGA产品开发工具、开发环境以及验证工具。如无要求，可由FPGA产品承制方自行选择开发工具。如果使用国产化器件，但编译等使用的是原厂环境，应在本章节进行说明。

## 2. 载人航天FPGA软件需求分析



### ■ 详细需求-接口及时序需求

- 提出接口定义、管脚绑定、时序要求等要求
- 根据研制任务书要求分章节详细描述FPGA产品与周边器件的接口
- 明确信号名称、信号标识、信号方向、管脚分配、电平标准、初始化状态、接口时序要求、未使用管脚的处理方式等，同时给出配置项对各外设接口的操作要求，结合器件手册和具体电路进行时序需求分析
- 接口要求一般包括：总线协议、遥测参数格式定义、上行注入数据格式等内容
- 对可编程要求、地址空间分配以及接口的收发能力等进行分析

## 2. 载人航天FPGA软件需求分析



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study,UCAS

### ■ 详细需求-功能需求

- 提出实现功能、时序要求、协议算法、地址空间分配、软件可编程要求、操作方式、IP 核复用等要求
- 对FPGA产品的各项功能进行需求分析。逐项叙述对FPGA产品所提出的功能要求，说明输入信号、处理方式、输出信号、明确各项功能发生的时间或者条件
- 对于复杂的系统，应该采取层次化的结构对功能进行逐次分解和描述
- 对功能需求进行标识和独立编号，便于追踪
- 需求描述应注意无歧义、易于理解、可测试、文文一致

## 2. 载人航天FPGA软件需求分析



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

### ■ 详细需求-性能需求

- 如工作时钟频率、EDAC产生校验字时间等要求
- 空间要求：明确数据可占用的最大存储空间
- 精度要求：确定FPGA软件的精度要求，如数据或数值计算的精度、数据传输的精度等
- 时间特性需求：确定软件的时间要求，如处理时间，器件之间的通讯和同步要求、指令间隔等



## 2. 载人航天FPGA软件需求分析

### ■ 详细需求-可靠性和安全性要求

- 抗SEU设计：对于任务书中选择的本产品的防护技术进行细化，明确如何实现该配置项的抗SEU措施。一般包括：单位/双位错的处理措施、关键数据和重要数据的存储与恢复等。如果有三模冗余需求，应描述需进行三模的模块、方式、工具选择
- 降额设计：包括设计资源降额和最大时钟频率，载人航天降额设计一般为80%
- 错误情况处理：分析系统可能的故障模式，并给出相应的应对措施
- 接口信号状态：明确FPGA产品在上电阶段和复位阶段接口信号的电平状态
- 在轨维护设计：明确FPGA产品在轨维护设计要求
- 其他可靠性设计：复位设计、同步设计、跨时钟域设计、IP核使用情况等

## 2. 载人航天FPGA软件需求分析



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

### ■ 其他要求

- 需要在需求规格说明里明确的任务书里的其他要求，如编程语言约定、文档要求等
- 编程语言约定：航天五院、八院、一院、科工集团、国军标
- 文档要求：验收的文档及配置管理等



## 2. 载人航天FPGA软件需求分析

### ■ 对照需求分析检查表查看任务是否明确

#### 一般要求

- 1、设计所遵循流程是否明确？（参考921 《FPGA软件研制技术要求》）
- 2、FPGA产品安全等级是否明确？（A、B、C、D 四级）
- 3、FPGA产品代号是否明确？
- 4、FPGA器件等级是否明确？
- 5、需求分析阶段对所选开发工具的可用性进行检查。

#### 功能需求

- 1、清楚地说明了单机任务书对FPGA要求的功能？
- 2、用到的协议、算法是否明确？
- 3、复位后内部寄存器的状态是否明确？
- 4、若FPGA是在上电加载后才加时钟信号，设计是否对该种相关的初态寄存器进行再次初始化，是否避免由该外部工作时钟初始不稳定引起本单机其他非相关功能异常现象？
- 5、FPGA与处理器数据接口方法和协议的描述是否明确，缓存器访问算法设计是否能避免数据溢出？
- 6、是否存在定点量化影响性能的FPGA产品，要开展全流程定点仿真？
- 7、有无不可实现的功能？

## 2. 载人航天FPGA软件需求分析



### ■ 对照需求分析检查表查看任务是否明确

#### 性能要求

- 1、供电电压是否明确？
- 2、功耗要求是否明确？
- 3、时钟运行频率、占空比要求是否明确？
- 4、业务模块性能指标是否明确？（信噪比、误码率）

#### 接口需求

- 1、清楚地说明了FPGA与外部的各有关接口关系？
- 2、清楚地说明了各有关接口的特性？
- 3、接口时序参数是否明确（建立延迟时间、输出延迟时间）？
- 4、是否说明了复位后接口信号的状态？

#### 可靠性和安全性需求

- 1、设计资源余量是否明确？
- 2、时钟余量是否明确？
- 3、未用管脚处理要求是否明确？
- 4、异常故障时如何处理是否明确？
- 5、在FPGA需求分析阶段，应细化单粒子防护方案，明确系统级单粒子防护技术，落实在FPGA需求分析报告中

## 2. 载人航天FPGA软件需求分析



### ■ 对照需求分析检查表查看任务是否明确

#### 可编程要求

- 1、在需求规格说明中明确提出编码实现应遵循的规范要求
- 2、可编程寄存器的初始值应当明确

#### 其他需求

- 1、计划进度要求是否明确?
- 2、IP核复用有无明确要求?
- 3、验收交付要求是否明确?

## 2. 载人航天FPGA软件需求分析



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

### (3) FPGA配置管理计划及质量保证计划

## 2. 载人航天FPGA软件需求分析



### ■ 配置管理计划要求

- 配置管理活动应贯穿于整个生命周期，保证FPGA软件在整个生命周期中的完整性和可追溯性
- 应通过配置标示、配置控制、配置状态纪实和配置审计，维护工作产品的完整性
- 应在项目策划阶段（需求分析阶段）指定配置管理计划，以指导FPGA软件生命周期内配置管理活动的开展，配置管理计划同开发计划、质量保证计划进行内部评审

### ■ 配置管理计划内容

- 明确FPGA产品配置管理活动的实施条件，规范整个型号中FPGA的配置管理活动，在整个研制生命周期中对配置项的变更进行跟踪，明确配置标识和基线划分，在软件研制过程中，建立并维护功能基线、分配基线和产品基线

## 2. 载人航天FPGA软件需求分析



### ■ 质量保证计划内容：

- 软件产品保证机构的任务和职责；
- 分配的资源，包括人员、设备等；
- 应编制的文档；
- 应遵循的标准、规范和约定，以及监督执行的措施；
- 需要进行的验证确认工作和实施规程；
- 软件配置管理和纠正过程的要求；
- 明确软件开发里程碑/基线时的产品保证；
- 软件可靠性要求保证活动；
- 支持软件产品保证的工具、技术和方法；
- 软件外协、外购、沿用要求；
- 风险控制要求；
- 软件质量与可靠性信息收集与传递要求。

## 2. 载人航天FPGA软件需求分析



### ■ 产品保证活动

- 对FPGA软件需求的**完整性、准确性、一致性、可验证性、易修改性和可追踪性评估**；
- 对FPGA软件设计输入方式和IP核资源进行**复核**；
- 对FPGA软件逻辑资源占用率、时钟余量等要求进行**确认**；
- 对FPGA软件安全性与可靠性需求进行**复核分析**，重点关注抗单粒子效应的需求；
- 对需求规格说明进行**正式评审**，对开发计划、配置管理计划、质量保证计划进行**内部评审**；
- 经过评审和批准的需求规格说明进入**受控库**，建立分配基线；
- 经过评审和批准的开发计划、配置管理计划、质量保证计划入**受控库**。

# 本节内容



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

## 3. 载人航天FPGA软件安全编程

### 3. 载人航天FPGA软件安全编程



#### ■ 符合编码规范

- 《A VHDL Modeling Guide TP-804》 美国海军VHDL设计指导
- 《Actel HDL Coding Style Guide》 Actel公司的编码规则
- 日本半导体技术学术研究中心 (STARC) 的VHDL、Verilog编码规范
- 华为的《VHDL代码书写规范技术规范》
- 航天、军工各系统，五院、八院、一院、科工集团航天器产品VHDL语言编程约定、  
航天器产品Verilog 硬件描述语言编程约定
- 军用可编程逻辑器件软件编程语言安全子集--VHDL语言篇
- 军用可编程逻辑器件软件Verilog语言编程安全子集

### 3. 载人航天FPGA软件安全编程



1、时钟设计类

2、复位和初始化

3、跨时钟域类

4、状态机类

5、可靠性安全设计类

6、编码规范类

7、接口设计类

8、EDA综合选项设置

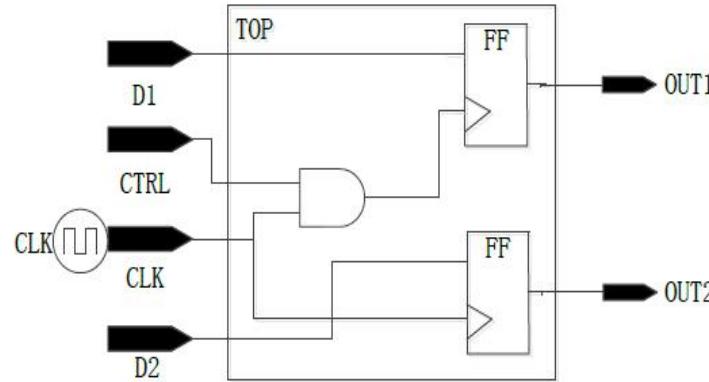


## 1、时钟设计类

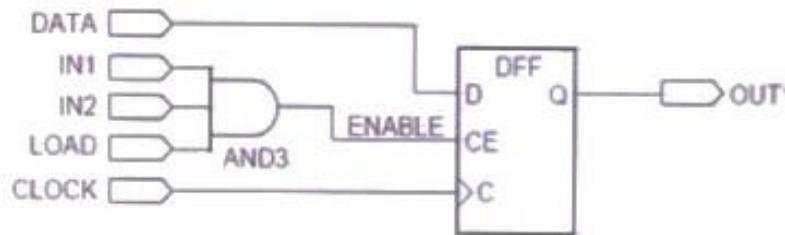
# 安全编程-时钟设计类



## 1、FPGA设计不应使用门控时钟



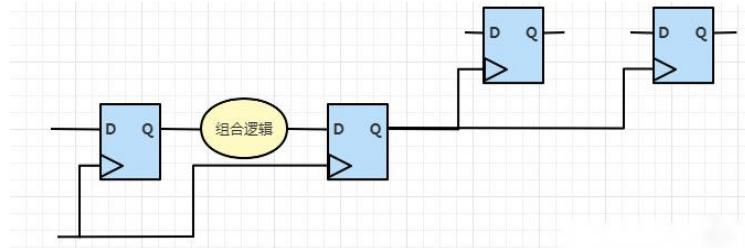
- 全局时钟 “clk” 通过了门控，进行时钟延迟和偏斜的微调会变的困难。在有低功耗需求必须对时钟进行控制时，应采用时钟使能的方式。
- FPGA内部时钟树来自外部输入时钟，尽管有时会通过FPGA内部组合逻辑将时钟连接至时钟树，但当组合逻辑控制存在问题时，会影响时钟延迟和偏斜的微调，也可能带来时钟信号上的毛刺。





## 2、不应该使用行波时钟

- 由计数器分频或状态机输出产生的时钟叫“**行波时钟 (ripple clock)**”。当前一级时序逻辑的数据输出被用作下一级时序逻辑的时钟输入时，这个时钟就是“**行波时钟**”



### ■ “行波时钟”的危害

- 行波行波时钟虽然不产生毛刺，但行波时钟使得与电路有关的时序计算变得复杂
- 远端电路时钟驱动能力减弱，使系统的工作不可靠
- **不推荐使用，必须使用时需要注意添加约束，且将行波时钟连接至全局时钟网络**

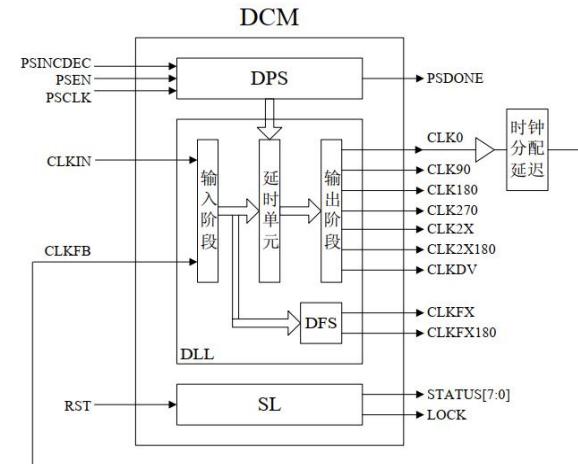


## 3、DCM或PLL IP核使用

■ 在满足器件应用环境要求的前提下，利用DCM或PLL能够有效提高FPGA时钟的驱动能力，使时序收敛

### ■ DCM或PLL使用原则

- 应权衡使用DCM或PLL，在辐射条件下DCM或PLL与FPGA其他资源相比更加脆弱；
- 为利于FPGA设计时序收敛，应合理规划使用时钟，尽量减少时钟数目；
- 为保证下DCM或PLL正常工作，检查外部端口输入的驱动时钟频率是否在下DCM或PLL工作的有效输入范围之内；
- 为保证下DCM或PLL的时钟抖动和稳定性，推荐将其设置为反馈模式。





### 3、DCM或PLL IP核使用

#### ■ 以下两种情况发生应将DCM或PLL复位

- DCM或PLL 的**LOCKED**信号变低，输出时钟未锁定；
- 用一个可靠稳定的时钟(DCM/PLL的输入时钟或本地时钟)对DCM或PLL的输出频率进行监控，当**输出频率异常**时；



## 2、复位和初始化



## 1、使用复位方式对寄存器进行初始化赋值

- FPGA内部所有信号（除无外部复位需用户上电自复位，使用到的寄存器），都需要使用复位方式对寄存器进行**初始化赋值**
- **案例：不确定信号量参与逻辑运算**  
**问题描述：**某FPGA配置项脉冲遥控信号在复位阶段高低电平不断跳变



## ■ 原因分析：

- 设计中对内部信号未进行初始化处理，导致布局布线后脉冲遥控信号在复位阶段呈现出**了不确定态**
- 某些FPGA器件（例如：ACTEL型反熔丝FPGA）上电后，其内部信号初态为**随机值**，当这些信号进行传递、参与运算时，应确保其值的确定性
- FPGA复位的目的是使FPGA内部寄存器和IO进入一个预先设计的状态，确保参与逻辑运算的信号量有确定的值、状态机进入预期状态、FPGA进入预期的功能



## 2、复位设计时要做到“异步复位，同步释放”

- 复位信号可以与时钟信号不同步，将复位信号的释放与时钟信号同步

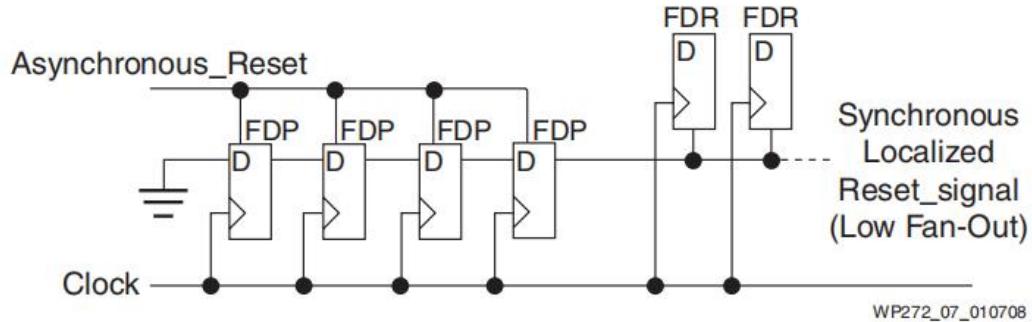


Figure 7: Localized Reset



### 3、不应将同一个复位信号同时作为同步复位和异步复位使用

- 复位信号不能在一个寄存器用于同步复位，在另一个寄存器用于异步复位
- 一个复位信号同时作为同步复位和异步复位，综合时可能出现非正常结果
- 同时包含异步复位和同步复位在逻辑综合和布局布线中也可能导致其他问题



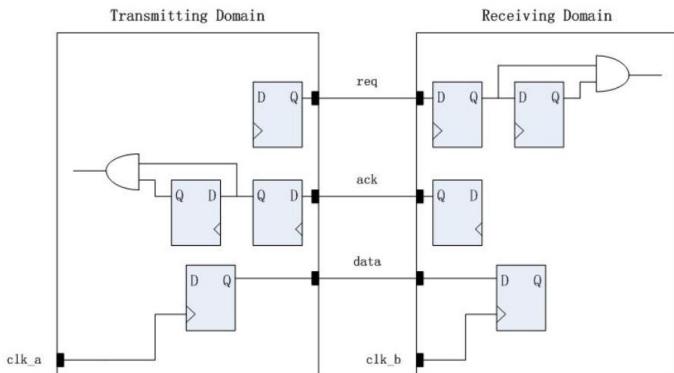
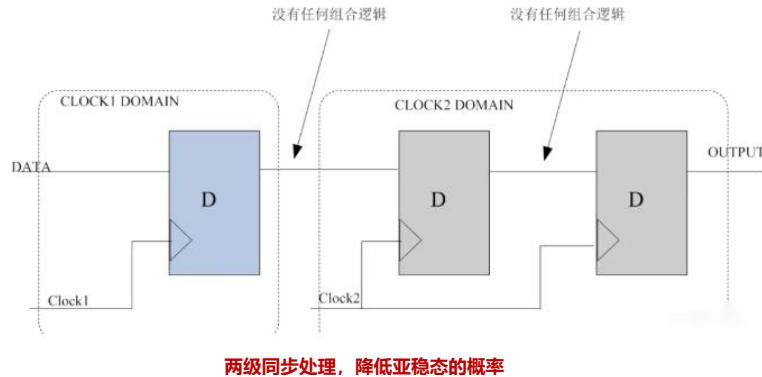
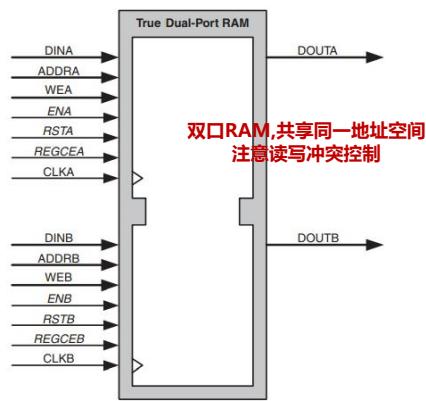
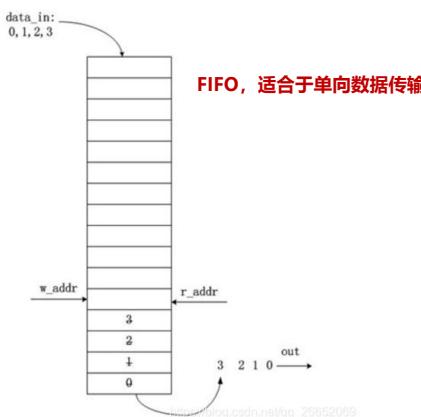
## 3、跨时钟域类

# 安全编程-跨时钟域类



## 1、注意跨时钟域信号的处理

- 跨时钟域的时钟分类：慢速时钟->快速时钟，快速时钟->慢速时钟
- 跨时钟域的信号分类：单bit数据，多bit数据
- 单bit数据，慢速时钟->快速时钟：采用两级同步处理、握手机制
- 单bit数据，快速时钟->慢速时钟：采用握手机制
- 多bit数据，慢速时钟->快速时钟：FIFO、双口RAM、握手机制
- 多bit数据，快速时钟->慢速时钟：FIFO、双口RAM、握手机制



1. 发送域将data有效;
2. 发送域将req信号有效, 通知接收域读取信号;
3. 接收域读取数据后, 接收域发送ack信号, 通知发送域读取结束



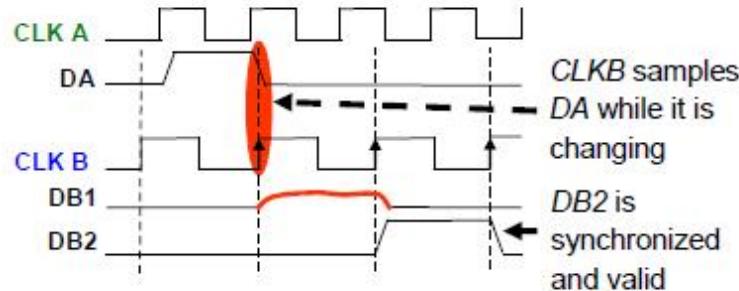
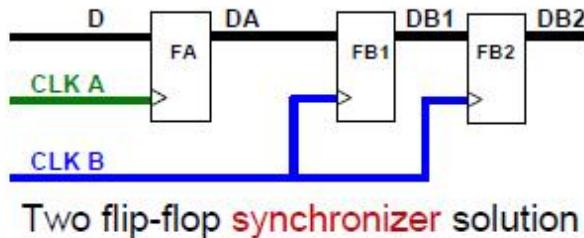
## ■ 案例：数据未进行正确同步

- **问题描述：**高精压传联试时，发现频率值存在跳变的异常现象



## ■ 原因分析

- 如下例所示，设计中对异步信号DA处理不当，锁存1次的DB1是一个亚稳态数据，将DB1与锁存2次的DB2进行运算供后级电路使用，导致亚稳态传播，造成各级触发器锁存值不一致，引发时序差异。



# 安全编程-跨时钟域类



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

- **纠正措施：**对输入的DA信号先锁存2次，再进行滤波后使用
- **风险控制要求：**在满足时序要求的情况下，对于外部的异步输入信号，要进行2级同步化处理



## 2、注意异步时钟相位关系的处理

- 同一个工程中，两个异步时钟域存在数据交互时，两个异步时钟的相位关系应保持固定，或定期对相位误差进行修正
- **案例：异步时钟相位不固定**  
**问题描述：**某信号处理FPGA软件时钟控制字设置错误

# 安全编程-跨时钟域类



## ■ 原因分析：

- FPGA中存在两个来自不同时钟源的时钟clk1,clk2，在FPGA内部分别设置两个时钟的计数器a和计数器b，由于两个时钟域的信号不同步，无法保持两个时钟的固定相位关系，导致经过较长时间形成误差累计，最终导致输出数据错误
- 如下图所示，刚开始两个计数器的计数正确，随着时间的延长误差累计，当计数器a计到853时，计数器b计到844，输出数据错误



# 安全编程-跨时钟域类



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

- **纠正措施:** 分析时钟间的相位关系, 定期对误差进行修正
- **风险控制要求:** 同一设计中使用多时钟时, 要充分考虑时钟间的同步关系



## 4、 状态机类



## 1、状态机需设置**default/others**状态

- 状态机需设置**default (Verilog) /others (VHDL)** 状态，且从**default/others**状态可以跳转到空闲状态，防止状态机跳转到异常状态导致状态机死锁，FPGA功能中断
- 需要对异常状态应进行正确处理，并对代码进行编码规则检查，避免发生状态机死锁



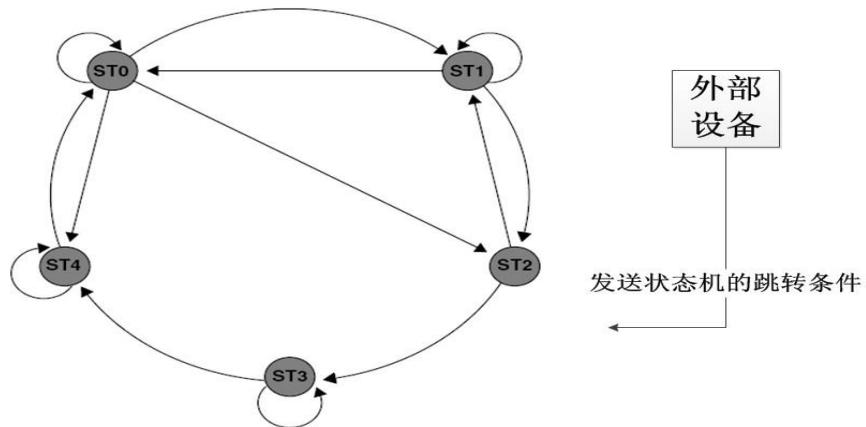
## 2、状态机设置超时保护防止死锁

- 状态机的跳转依赖于外部输入的信号时，则需要考虑该信号失效的情况，防止状态机接收不到有效信号一直停留在该状态，导致功能中断
- **案例：**状态机长时间等待跳转条件导致死锁  
**问题描述：**状态机跳转仅依赖于外部信号，外部信号出现问题时，状态机发生了死锁



## ■ 原因分析：

- FPGA对外部AD芯片进行控制时采用状态机方式实现，状态机对AD芯片的反馈信号进行判断，只有它出现上升沿时状态机才会跳转。当外部信号发生故障时，状态机一直等待，进入死锁无法恢复。





- **纠正措施：**对外部信号的判断增加了超时处理避免死锁发生
- **风险控制要求：**对状态机的外部输入信号进行必要的超时处理，避免由外部故障引发的状态机死锁



### 3、设置状态机安全模式

- 在写状态机时，会添加default语句（VHDL是others）分支，当出现异常状态时，保证状态机能跳转到正常状态
- 但是部分EDA工具（例如Quartus,synplif等EDA工具）会对default语句进行有意的自动忽视，这是对状态机进行优化的结果。为了安全起见，在航天器产品上需要**设置安全状态**，保证当出现异常状态时，状态机可以跳转到正常的状态



## 3、设置状态机安全模式

### ■ 设置方法

- ISE: 综合选项Safe Implementation中选择Yes;
- Vivado:

|                           |      |
|---------------------------|------|
| FSM Encoding Algorithm    | Auto |
| Safe Implementation       | Yes  |
| Case Implementation Style | None |

方式一：在XDC文件中添加约束的方式进行状态机安全模式设置

```
set_property FSM_SAFE_STATE reset_state [get_cells -hier -filter NAME=next_state]
```

方式二：在VHDL文件中添加综合属性fsm\_safe\_state设置状态机安全模式；

```
attribute fsm_safe_state : string;
attribute fsm_safe_state of next_state : signal is "auto_safe_state";
```



## 5、可靠性安全设计类



## 1、地址信号译码正确性防护

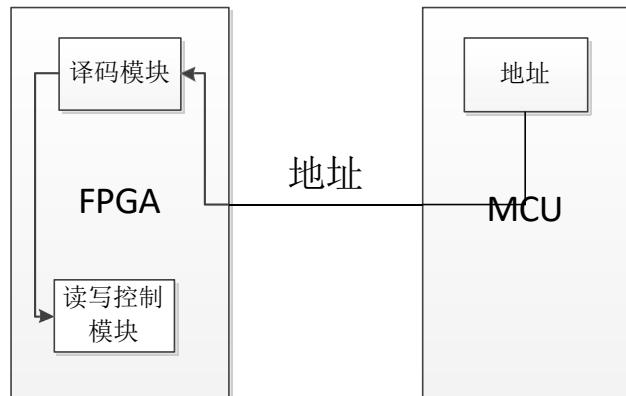
- FPGA对于外部输入的地址信号，要对全地址字段进行判断，并考虑地址译码可能出现的竞争问题
- **案例：**FPGA对错误的地址信号进行了响应操作  
**问题描述：**某型号FPGA配置项进行接口测试时发现，该配置项对MCU发来的错误地址信号进行了响应操作，导致读写操作错误

# 安全编程-可靠性安全设计类



## ■ 原因分析

- 在FPGA与MCU的接口中，FPGA对MCU的地址信号进行译码，并根据译码结果对FPGA进行相应的读写操作。设计人员认为地址信号由MCU保证，不会发送未定义的地址范围，因此FPGA对MCU地址信号进行译码时仅判别部分字段，导致FPGA对错误的地址信号译码为有效，并进行了读写操作。



FPGA与MCU接口示意图

# 安全编程-可靠性安全设计类



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

- **纠正措施：**FPGA设计中应对MCU的全地址字段进行判
- **风险控制要求：**设计中要考虑地址译码可能出现的竞争



## 2、主备份切换考虑毛刺等异常情况

- FPGA在进行主备份切换设计时，不能想当然谁有数据到达就切换到该单机模式下，在确定主备份中断处理机制时，充分考虑中断冲突和由毛刺引起的中断误判问题



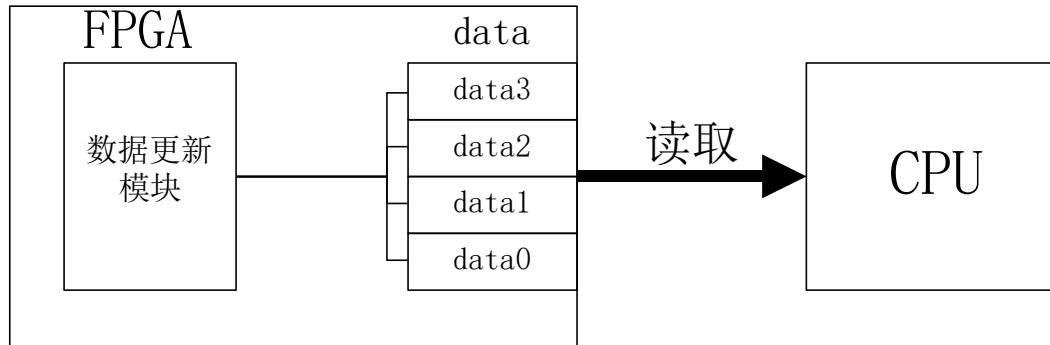
### 3、避免拼帧问题

- FPGA同步收发数据时，对读数据进行锁存，避免多字节数据的读操作出现拼帧现象
- **案例：**对于多个字节的读操作，读出的前一半为旧数据，后一半为新数据  
**问题描述：**某型号进行测试时发现，该配置项CPU读取的数据前一半为旧数据，后一半为新数据，出现拼帧现象



## ■ 原因分析

- 一个数据由多字节组成的，需要分多次读取，FPGA没有对该数据进行锁存处理。在数据读到一半时，数据缓冲区更新，导致CPU读取的数据前一半为旧数据，后一半为新数据，出现拼帧现象：



CPU读取的数据示意图

# 安全编程-可靠性安全设计类



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

- **纠正措施：**FPGA对整个数据进行锁存，保证在读取数据的过程中，数据更新不会对此次读取数据产生影响
- **风险控制要求：**对读数据进行锁存，避免多字节数据的读操作出现拼帧现象



## 4、避免判决条件不完善

- FPGA设计中存在很多判断语句，在进行判决时，要充分考虑判决条件的完善和逻辑充分性，避免因为判决方式不合理而导致的异常

## 5、关键存储器进行三模冗余和自刷新设计

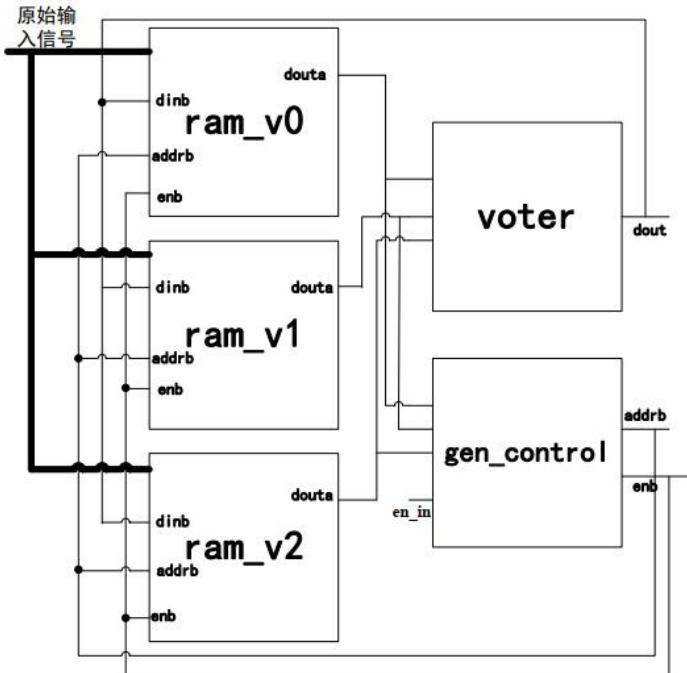
- 对选取的关键存储器进行三模冗余，将判决后结果送出给其他模块使用，同时将判决结果反馈给存储器输入部分进行纠错

# 安全编程-可靠性安全设计类



## ■ 三模冗余

- 对选取的存储器进行三模冗余，将判决后结果送出给其他模块使用，同时判断三个冗余域的结果，一旦有一个冗余域出现错误，利用判决后结果对存储器存储的数据进行更新，对错误进行纠正。





## 6、编码规范类



## 1、避免生成锁存器

- FPGA设计中应避免因编码问题生成锁存器，导致系统稳定性下降。
- 案例：无法实现对输入信号的毛刺滤除

**问题描述：**某FPGA配置项外部信号的毛刺直接传递到了输出端口，影响了FPGA与外设的交互



## ■ 原因分析

- 组合逻辑、锁存器等电平触发的单元，都容易存在竞争冒险，输入信号的毛刺会导致非预期的错误，该设计中使用了不合理的的设计结构，后端实现成为了锁存器，降低了对毛刺的过滤能力，导致问题发生。
- 例如下列代码，产生了锁存器，进一步查询综合报告，有如下描述：

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity latch_test is
 port (din : in std_logic_vector (7 downto 0);
 dout : out std_logic_vector (7 downto 0));
end latch_test;

architecture Behavioral of latch_test is
begin
 process (din)
 begin
 if (din >X"7F") then
 dout <= din;
 end if;
 end process;
end Behavioral;
```

```
HDL Synthesis Report

Macro Statistics
Latches : 1
8-bit latch : 1
Comparators : 1
8-bit comparator greater : 1

* Advanced HDL Synthesis

Loading device for application Rf_Device from file 'v300.nph

Advanced HDL Synthesis Report

Macro Statistics
Latches : 1
8-bit latch : 1
Comparators : 1
8-bit comparator greater : 1
```



## ■ 锁存器的危害

- 对毛刺敏感，不能异步复位，所以上电后处于不稳定的状态
- LATCH是组合逻辑器件，电平触发的存储器，会使静态时序分析变得非常复杂
- 补全判断条件可以消除锁存器



## 2、避免数据总线读写冲突

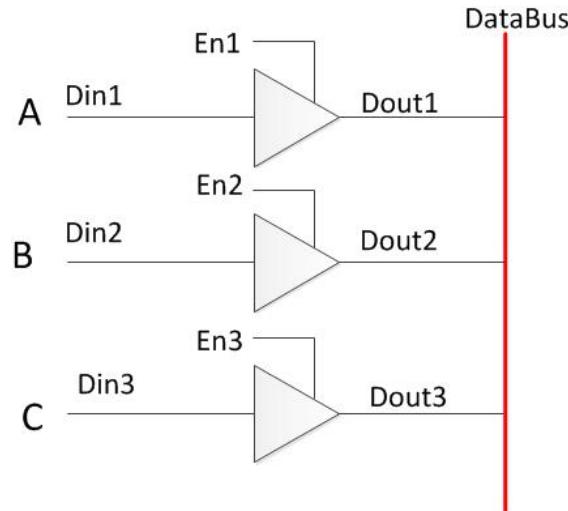
- FPGA设计中应避免因编码问题导致多路数据同时向数据总线上写数据，导致总线数据冲突
- 案例：控制信号切换不同步导致总线数据冲突

**问题描述：**设计中的内部模块使用了三态逻辑，并且以总线形式与其它模块进行连接。各个模块之间的三态门使能信号切换不同步导致了短时间的短路



## ■ 原因分析：

- 如图所示，若A模块的使能信号En1有效时，Din1和Dout1直通，此时En2和En3使能无效时，Dout1向总线写数据。但是若此时使能信号En2和En3其中任何一个也有效，则总线上数据冲突。应添加相应的控制信号，使某一时刻只有一个使能信号有效：





### 3、注意条件语句优先级问题

- FPGA设计中if...else语句是有优先级的，第一个if具有最高优先级，最后一个else优先级最低。最高优先级的电路靠近电路的输出，输入到输出的延时较短；最低优先级的电路远离输出端，输入到输出的延时较长。
- **案例：赋值操作不合理**  
**问题描述：**某FPGA配置项下传图像存储时，像素发生错位现象。

# 安全编程-编码规范类



## ■ 原因分析

- 同一进程中，分别有两处条件满足后，会对同一信号进行赋值操作，两个条件都满足时，以最后一次赋值有效。由于设计时未考虑两个条件的判断顺序，导致下传图像偶尔会有错位现象发生。

```
architecture Behavioral of test_fz is
begin

process(clk)
begin
 if(clk'event and clk='1')then
 if(a>=X"a")then
 dout<='1';
 else
 ce<='0';
 end if;

 if(b>=X"e")then
 dout<='0';
 else
 ce<='0';
 end if;
 end if;

end process;

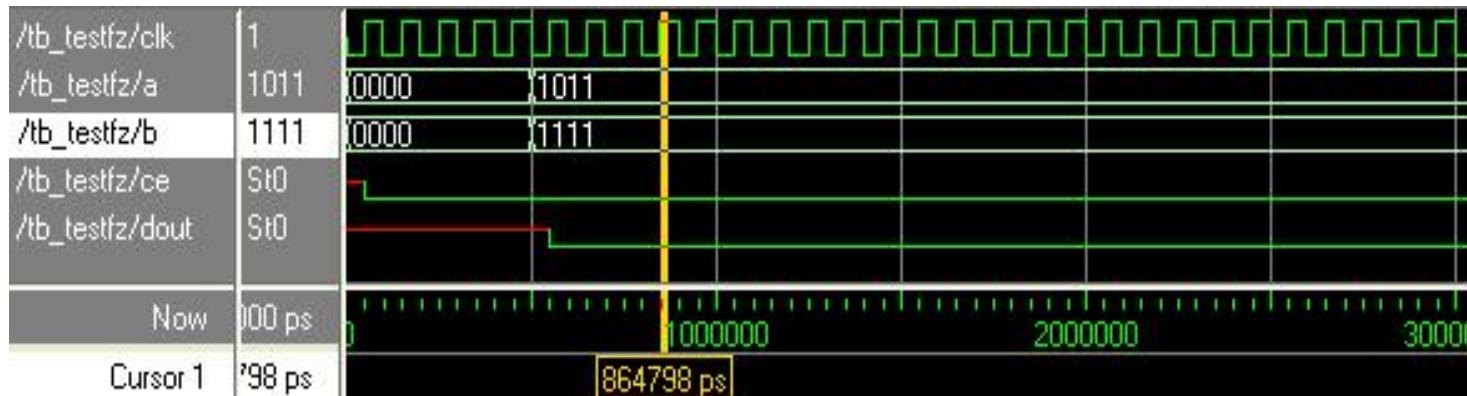
end Behavioral;
```

# 安全编程-编码规范类



## ■ 原因分析

- 如下示例：同一process中，两个无优先级的if条件都满足时，以最后一次赋值有效。



# 安全编程-编码规范类



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

- **纠正措施：**调整信号的赋值条件判断顺序
- **风险控制要求：**同一进程对同一信号的赋值操作，应结合功能要求设计正确的赋值顺序



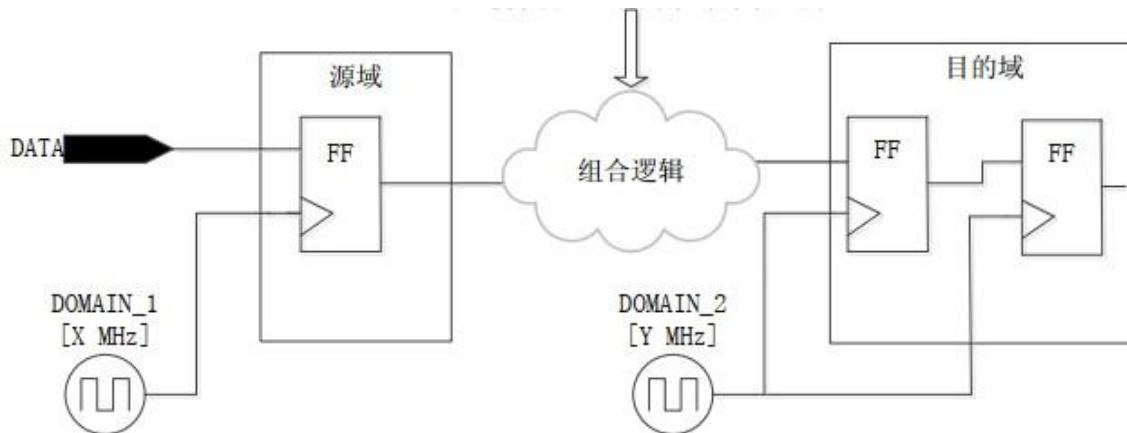
## 4、组合逻辑不应置于两个时钟域之间

- FPGA工程中信号跨时钟域处理时，不应在两个时钟域中间放置组合逻辑，应先进行两级同步化处理/Fifo、RAM缓存处理，然后再进行组合逻辑运算。
- 案例：两个时钟域间存在组合逻辑  
**问题描述：**某FPGA配置项偶尔出现一个时钟周期的数据错误



## ■ 原因分析

- 使用同步电路解决跨时钟域信号处理问题时，为保证其正常工作，跨时钟域的信号从源域至同步电路的第一个触发器间不应经过任何组合逻辑。同步电路的第一个触发器对组合电路产生的短时脉冲波形干扰敏感，如瞬时脉冲干扰恰好满足建立保持时间需求，将传播一个伪有效值至目的域的下游逻辑。



# 安全编程-编码规范类



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

- **纠正措施：**将组合逻辑放置两级同步化寄存器之后
- **风险控制要求：**FPGA编码设计时，避免能引起FPGA内部寄存器亚稳态的问题，避免将组合逻辑置于两个时钟域之间



## 5、赋值表达式右侧的所有信号均应列在敏感列表中

- 所有出现在赋值语句右侧和条件表达式的判断信号应被定义在组合进程块的敏感列表中，可避免 RTL和综合后仿真的差异
- 案例：敏感列表未列出所有信号  
**问题描述：**某FPGA配置项出现前仿真和后仿真结果不一致



## ■ 原因分析

- 输入信号D1、D2出现在敏感列表中，条件判决信号未列在敏感列表中，造成前仿真和综合后仿真结果不一致。

```
process(D1, D2) begin
 if (SEL = '0') then
 Q <= D1;
 else
 Q <= D2;
 end if;
end process;
```



## ■ 原因分析

- 敏感信号列表出现在always或process块中，其典型行为级的含义是：只要敏感信号列表内的信号发生电平变化，则always或process语句就执行一次，因此设计人员必须将所有的输入信号和条件判决信号都列在信号列表中。不完善的信号列表会造成不同的仿真和综合结果。在实际的FPGA开发中，EDA工具都会默认将所有的输入信号和条件判决信号作为触发信号，增减敏感信号列表中的信号不会对最终的执行结果产生影响，会使仿真结果不一样，从而影响设计人员的仿真测试。

# 安全编程-编码规范类



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

- **纠正措施：**将输入信号和条件判决信号都列在信号敏感列表
- **风险控制要求：**FPGA编码设计时，将所有的输入信号和条件判决信号都列在always或process信号敏感列表中，避免综合前仿真出现异常



## 7、接口设计类



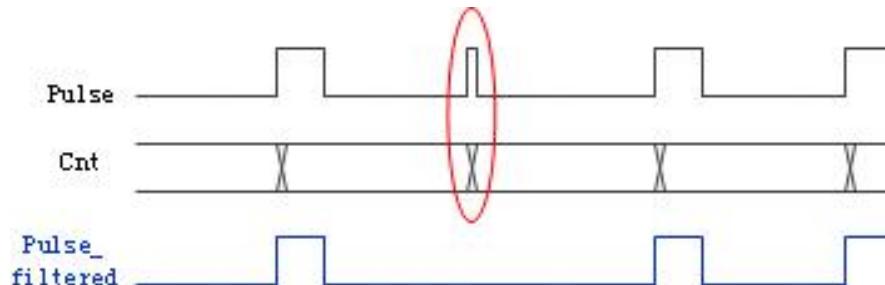
## 1、外部输入的脉冲信号进行滤毛刺设计

- FPGA外部输入的脉冲信号，要进行滤毛刺设计，防止因毛刺导致接口控制或接口数据异常
- 案例：外部输入的脉冲信号未进行滤毛刺设计  
**问题描述：**某配置项DSP从FPGA读取的脉冲计数值错误，发现转速偶尔不正常



## ■ 原因分析：

- FPGA对输入的转速脉冲进行计数并供DSP读取，FPGA设计中没有考虑外部输入信号存在的毛刺，直接将外部输入的转速脉冲用作脉冲计数，导致计数结果存在偏差

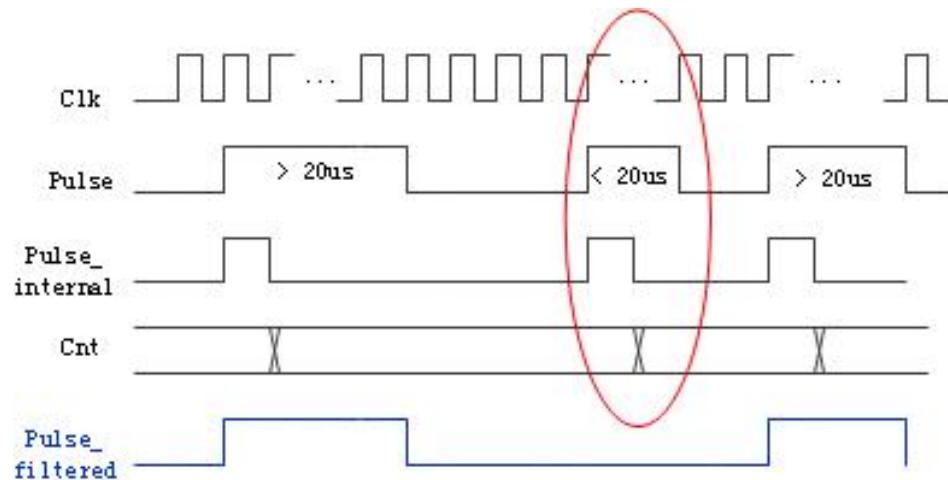


直接以脉冲作为时钟计数



## ■ 原因分析：

- 对于某些对脉冲宽度有要求的设计，也需要滤除宽度不足的脉冲，  
设计中可以产生时钟宽度的脉冲以便计数



# 安全编程-接口设计类



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study,UCAS

- **纠正措施：**对外部输入的脉冲信号进行防抖动的滤毛刺处理
- **风险控制要求：**对外部输入的脉冲信号进行滤毛刺设计，避免错误计数



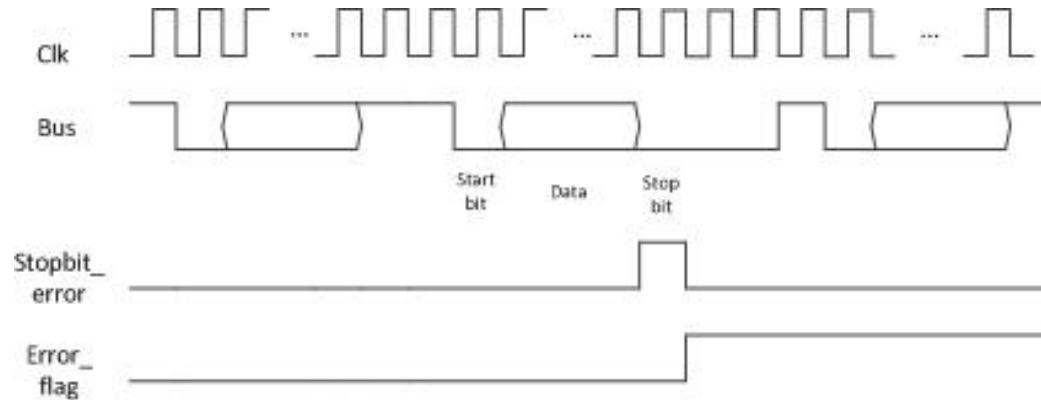
## 2、与接口相关标识信号状态及时更新

- FPGA内部用于检测外部接口状态的信号，需要根据外部接口的变化及时更新，避免因内部信号更新不及时导致的接口数据异常
- **案例：内部错误标志未及时清除**  
**问题描述：**接收串口数据时，某字节的停止位错误，此后接收的所有数据均被丢弃。



## ■ 原因分析：

- 通讯协议要求，对所有异常的帧格式进行有效判断和处理，保证出错后不影响后续数据接收。FPGA接收串口数据时，接收到的某一字节数据停止位是错误的，FPGA将停止位错误标识置为有效后，未及时置回到无效状态，导致后续接收到的所有数据都被丢失。



# 安全编程-接口设计类



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

- **纠正措施：**等待固定时刻后清除错误标识位，重新开始数据帧的接收和判断。
- **风险控制要求：**对接口通信中的重要标识进行及时处理，避免影响后续通讯。



### 3、接口时序满足器件要求

- FPGA设计时应满足外部器件的接口时序要求，尤其对于 DPS/CPU+FPGA+AD/FLASH的硬件设计，往往数据/地址总线直连，控制信号通过FPGA到各模块，要保证经过FPGA延迟的控制信号与直连的数据/地址总线时序满足 AD/FLASH的时序要求
- 案例：未考虑外围器件时序

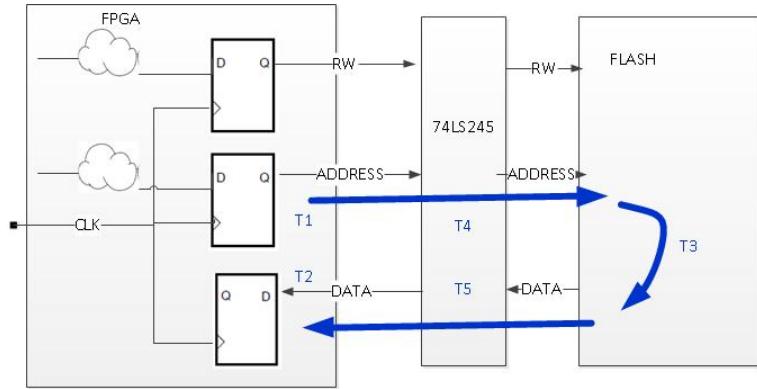
问题描述：某配置项在最大工况下进行仿真，发现从FLASH中读出的数据是“XX”

# 安全编程-接口设计类



## ■ 原因分析

- FPGA与FLASH之间的输入输出数据都经过了74LS245芯片，来回约有20ns的延时，而设计师未考虑信号经过驱动芯片的来回延时，导致从FPGA给FLASH读信号到FPGA收到FLASH经过245驱动芯片给出的数据之间最大延时为55ns，再加上布局布线后的延时影响，造成FPGA与FLASH接口时序中的建立/保持时间过于紧张，读取数据错误。



时序分析需要考虑从FPGA输出，  
经过外围器件，再返回FPGA的  
全路径的延时。

# 安全编程-接口设计类



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study,UCAS

- **纠正措施：**对FPGA与FLASH的接口进行再设计，确认在芯片延时及最大工况的影响下，可以满足FLASH接口时序要求
- **风险控制要求：**充分考虑外围芯片的延时，避免时序紧张



## 8、EAD综合选项

# 安全编程-EAD综合选项



國科大杭州高能研究院  
Hangzhou Institute for Advanced Study, UCAS

综合选项的设置对综合结果有着潜在的影响，根据  
EDA软件说明，合理的综合选项设置为FPGA软件  
的安全编程保驾护航。

# 安全编程-EAD综合选项



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study,UCAS

- 设置FSM（有限状态机）编码约束，确定使用的有限状态机的**编码方式**，按照航天器FPGA的研制规范，应选择**One-hot**，抗SEU的可靠性更高
- 使用**安全模式**来实现FSM
- 按照航天器FPGA的研制规范，根据SRAM型FPGA单粒子防护效应要求，**避免生成分布式RAM**

# 本节内容



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study,UCAS

## 4. 载人航天FPGA软件测试验证

## 4. 载人航天FPGA软件测试验证



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study,UCAS

### (1) 功能验证



## 4. 载人航天FPGA软件测试验证

- 设计**代码规则检查**, 编码规则符合硬件描述语言约定
- **人工走查**, 检查重点应包括可靠性设计、抗单粒子反转效应设计等
- 在EDA平台下完成**功能仿真**, 对设计的功能进行验证
- 完善**验证代码和验证用例**
- **仿真验证报告相关内容的编写**

# 4. 载人航天FPGA软件测试验证



## ■ 常见规则检查工具

| 序号 | 公司       | 软件名称     |
|----|----------|----------|
| 1  | Cadence  | HAL      |
| 2  | Synopsys | Leda     |
| 3  | Atrenta  | SpyGlass |
| 4  | Novas    | nLint    |

## 4. 载人航天FPGA软件测试验证



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

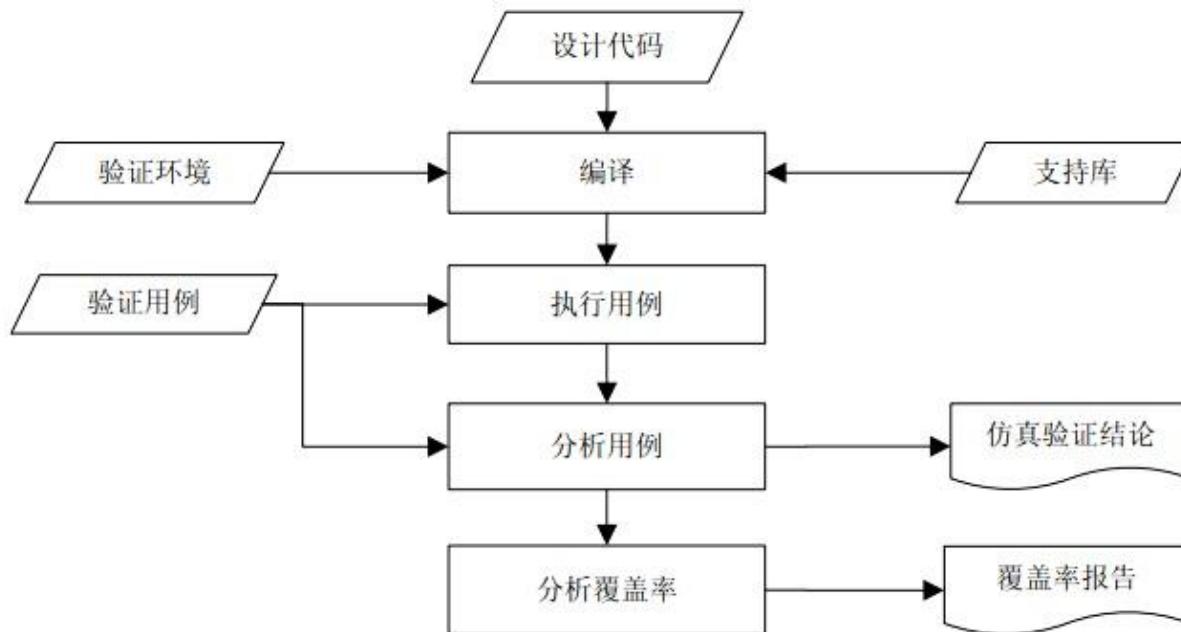
### ■ 人工走查的重点

- 跨时钟域走查
- 状态机设计走查
- 可靠性设计走查
- 单粒子效应防护设计走查

## 4. 载人航天FPGA软件测试验证



### ■ 仿真验证流程



## 4. 载人航天FPGA软件测试验证



### ■ 常见仿真工具：

| 序号 | 公司       | 软件名称                                |
|----|----------|-------------------------------------|
| 1  | Cadence  | IES (Incisive Enterprise Simulator) |
| 2  | Synopsys | VCS (Verilog Compiled Simulator)    |
| 3  | Mentor   | Modelsim、Questasim                  |
| 4  | Xilinx   | ISim                                |

## 4. 载人航天FPGA软件测试验证



### ■ 仿真工具覆盖率指标

不同工具覆盖率指标不同，覆盖率既相似，又有区别

| 工具名称      | 语句、分支   | 条件、表达式  | 状态机状态 |
|-----------|---------|---------|-------|
| QuestaSim | 语句、分支   | 条件、表达式  | 状态机状态 |
| ModelSim  | 语句、分支   | 条件、表达式  | 状态机状态 |
| VCS       | 行覆盖率、分支 | 表达式     | 状态机状态 |
| IES       | 块覆盖     | 表达式状态覆盖 | 状态机状态 |

## 4. 载人航天FPGA软件测试验证



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

### ■ 载人航天FPGA软件研制技术要求

- 对于A、B级FPGA应统计语句、分支、条件、表达式和状态机覆盖率，根据覆盖率完善验证代码和验证用例，语句、分支、条件、表达式和状态机覆盖率应达到100%。对覆盖率达到要求的FPGA应对未覆盖的部分逐一进行分析和确认。

## 4. 载人航天FPGA软件测试验证



### ■ Modelsim覆盖率定义

- **语句覆盖率 (Statement coverage)** : 统计代码行中每条语句的执行情况。
- **分支覆盖率 (Branch coverage)** : 对if、case等语句块中，每个分支的执行情况，进行统计。
- **条件覆盖率 (Condition coverage)** : 计算代码条件语句中“if”的条件执行情况。 (当if中语句真值表情况大于2时)。
- **表达式覆盖率 (Expression coverage)** : 计算代码中赋值语句的执行情况。 (需要考虑等号右边表达式的所有赋值情况)。
- **状态机覆盖率 (FSM coverage)** : 计算有限状态机的所有状态、转移和路径的覆盖情况。



## 4. 载人航天FPGA软件测试验证

|   |                |    |                                    |
|---|----------------|----|------------------------------------|
| ✓ | ✓              | 39 | process(clk,rst)                   |
| ✓ | ✓              | 40 | begin                              |
|   |                | 41 | if(rst = '0')then                  |
|   |                | 42 | dout_i <= (others => '0');         |
|   | X <sub>T</sub> | 43 | elsif(clk'event and clk = '1')then |
|   | X <sub>S</sub> | 44 | if(a >= x"05") then                |
|   |                | 45 | dout_i <= x"06";                   |
|   |                | 46 | elsif(a <= x"02") then             |
|   | X <sub>T</sub> | 47 | dout_i <= x"10";                   |
|   | X <sub>S</sub> | 48 | else                               |
|   |                | 49 | dout_i <= x"01";                   |
|   |                | 50 | end if;                            |
|   |                | 51 | end if;                            |
|   |                | 52 | end process;                       |

|   |                |    |                                    |
|---|----------------|----|------------------------------------|
| ✓ | ✓              | 39 | process(clk,rst)                   |
| ✓ | ✓              | 40 | begin                              |
|   |                | 41 | if(rst = '0')then                  |
|   |                | 42 | dout_i <= (others => '0');         |
|   | X <sub>T</sub> | 43 | elsif(clk'event and clk = '1')then |
|   | X <sub>S</sub> | 44 | if(a >= x"05") then                |
|   |                | 45 | dout_i <= x"06";                   |
|   |                | 46 | elsif(a <= x"02") then             |
|   | X <sub>T</sub> | 47 | dout_i <= x"10";                   |
|   | X <sub>S</sub> | 48 | else                               |
|   |                | 49 | dout_i <= x"01";                   |
|   |                | 50 | end if;                            |
|   |                | 51 | end if;                            |
|   |                | 52 | end process;                       |

√ 为仿真覆盖分支  
✗ 为仿真未覆盖分支

## 4. 载人航天FPGA软件测试验证



### ■ Modelsim源代码窗口图标含义：

| 图标                                                 | 描述                     |
|----------------------------------------------------|------------------------|
| 绿色的对勾 <span style="color: green;">√</span>         | 本语句已被执行过               |
| 红色的 <span style="color: red;">X<sub>T</sub></span> | 真分支没有被覆盖 (BC column)   |
| 红色的 <span style="color: red;">X<sub>F</sub></span> | 假分支没有被覆盖 (BC column)   |
| 红色的 <span style="color: red;">X<sub>C</sub></span> | 条件没有被覆盖 (Hits column)  |
| 红色的 <span style="color: red;">X<sub>E</sub></span> | 表达式没有被覆盖 (Hits column) |
| 红色的 <span style="color: red;">X<sub>S</sub></span> | 语句没有被覆盖 (Hits column)  |
| 红色的 <span style="color: red;">X<sub>B</sub></span> | 分支没有被覆盖 (Hits column)  |
| 绿色的 <span style="color: green;">E</span>           | 本行被指定不做代码覆盖统计          |

## 4. 载人航天FPGA软件测试验证



### ■ 仿真验证注意事项

- **仿真工具优化**: 仿真工具软件在对验证环境和RTL代码进行编译时可对代码进行优化处理，是否进行优化处理可由开关控制属性控制
- **敏感信号错误**: 仿真验证时未在敏感列表中的输入信号发生变化，组合逻辑的输出不会响应，而导致仿真结果与设计预期不一致
- **多余敏感信号**: 在仿真时会造成process或always block意外进入，可能导致覆盖率信息不准确，因此要删除多余敏感信号
- **随机化数据**: 采用随机化数据激励可避免人工设计数据激励的片面性和局限性
- 覆盖率统计和分析的目的不是使覆盖率指标达到某个要求，而是为了**评估仿真验证的充分性**

## 4. 载人航天FPGA软件测试验证



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

### (2) 时序验证

## 4. 载人航天FPGA软件测试验证



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study,UCAS

### ■ 阶段任务

- 在EDA平台下完成静态时序分析
- 完善验证代码，在EDA平台下完成时序仿真，对设计的功能、时序进行验证

### ■ 技术要求：

- 时序分析应覆盖最大、最小、典型三种工况
- 应进行时序仿真和功能仿真结果的一致性比较，确保布局布线后的门级网表和 RTL代码逻辑等价性，也可采用逻辑等价性检查工具进行 RTL代码与门级网表的逻辑等价性检查

## 4. 载人航天FPGA软件测试验证



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

### ■ FPGA软件的时序验证-静态时序分析

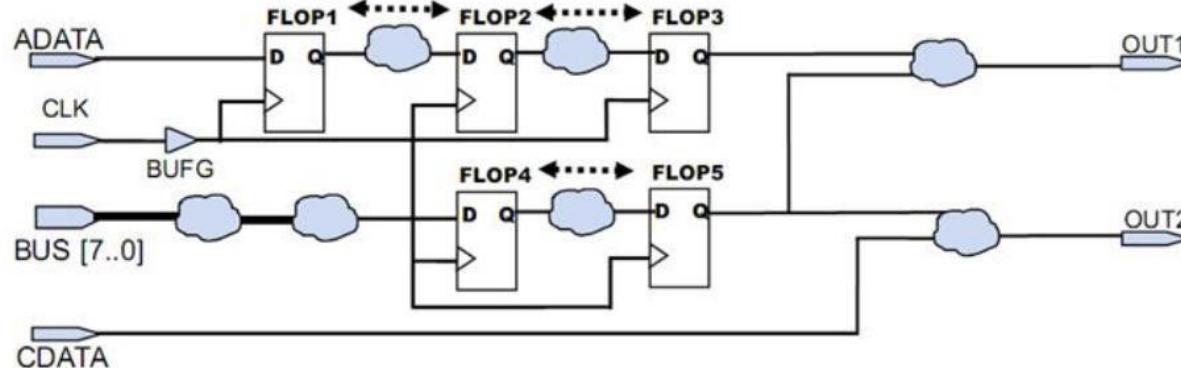
- 静态时序分析：是一种穷尽分析方法，它提取整个电路的所有时序路径，通过计算所有路径上的器件延时、传播延时来确定违背时序约束的路径；
- 静态时序分析在分析过程中计算时序路径上数据信号的到达时间和要求时间的差值，以判断是否存在违反设计规则的错误。

## 4. 载人航天FPGA软件测试验证



### ■ 要求

- 内部时序需求：内部所有寄存器应能满足其自身的**建立保持时间要求**，包括内部生成时钟驱动的寄存器等。
- 外部时序需求：在**给定输入时序下**FPGA能**正确采样外部输入数据或正确输出数据**
- 载人航天：**降额后**，内部所有寄存器应能满足其自身的**建立保持时间要求**，**无时序违例**





## 4. 载人航天FPGA软件测试验证

### ■ 工况

- 工况一般指：**制程**（Process），**电压**（Voltage），**温度**（Temperature）。FPGA设计所用的芯片是确定的，即制程是确定的，所以时序分析中所指工况一般指温度和电压。所有合法工况至少应包括**最大**（温度最高电压最低）、**最小**（温度最低电压最高）、**典型**（室温典型电压）三种工况。
- 静态时序分析应覆盖**最大、最小、典型**三种工况

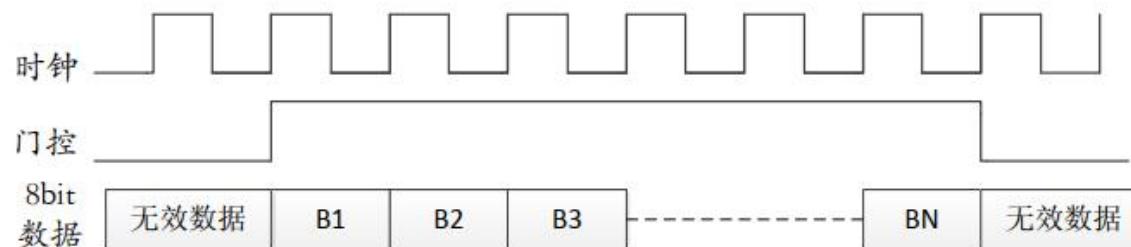
# 4. 载人航天FPGA软件测试验证



## ■ 案例

某型号FPGA任务书接口要求：

1. 通信双方发送时钟频率： 50MHz，余量 20%，占空比 40%-60%；
2. 发送端使用同步时钟上升沿发数，接收端采用同步时钟下降沿采样数据；
3. 发送端钟码关系：数据和时钟跳变沿滞后于时钟上升沿不大于 $\pm 3\text{ns}$ ；



## 4. 载人航天FPGA软件测试验证



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

### ■ 静态时序分析过程：

- 需求1为基本的时钟需求，可以使用Xilinx的PERIOD约束进行分析；
- 需求2为典型的外部输入情况，可以使用Offset In约束。

# 4. 载人航天FPGA软件测试验证



## ■ 时序分析：

- 给定条件：输入时钟以60MHz计（50M，20%余量），周期16.667ns，占空比40%- 60%。依次设置占空比为60%,50%,40%，进行分析。均满足建立保持时间要求。

| Period |            |                 |           |             |            |                 |
|--------|------------|-----------------|-----------|-------------|------------|-----------------|
| Find:  |            |                 |           |             |            |                 |
|        | Clock Pad  | * TIMESPEC Name | Period    | Pulse Start | Duty Cycle | Reference TIMES |
| 2      | CLK_LVDS_I | TS_CLK_LVDS_I   | 16.667 ns | High        | 60 %       |                 |

# 4. 载人航天FPGA软件测试验证



## ■ 时序分析

### 建立时间分析在最大工况下分析

```
TS_CLK_LVDS_I = PERIOD NET "CLK_LVDS_I_BUFGP/IBUFG" 16.667 ns HIGH 60%
 Setup paths
 3.347 From Inst_FPA_RX/LVDS_SYS_I_d2 (FF)
 3.347 From Inst_FPA_RX/LVDS_SYS_I_d2 (FF)
 3.470 From Inst_FPA_RX/LVDS_SYS_I_d2 (FF)
 Hold paths
 0.367 From Inst_FPA_RX/Inst_FIFO/U0/xst_fifo_generator/qconvfifo_rf/grf_rf/qmty
 0.375 From Inst_FPA_RX/Inst_FIFO/U0/xst_fifo_generator/qconvfifo_rf/grf_rf/qmty
 0.401 From Inst_FPA_RX/Inst_FIFO/U0/xst_fifo_generator/qconvfifo_rf/grf_rf/qmty
```

Slack (setup path): 3.347ns (requirement - (data path - clock path skew + uncertainty))  
Source: [Inst\\_FPA\\_RX/LVDS\\_SYS\\_I\\_d2 \(FF\)](#)  
Destination: [Inst\\_FPA\\_RX/Inst\\_FIFO/U0/xst\\_fifo\\_generator/qconvfifo\\_rf/grf\\_rf/qmty](#)  
Requirement: 6.667ns  
Data Path Delay: 3.315ns (Levels of Logic = 1)  
Clock Path Skew: 0.030ns (0.565 - 0.535)  
Source Clock: CLK\_LVDS\_I\_BUFGP falling at 10.000ns  
Destination Clock: CLK\_LVDS\_I\_BUFGP rising at 16.667ns  
Clock Uncertainty: 0.035ns  
  
Clock Uncertainty: 0.035ns  $((TSJ^2 + TIJ^2)^{1/2} + DJ) / 2 + PE$   
Total System Jitter (TSJ): 0.070ns  
Total Input Jitter (TIJ): 0.000ns  
Discrete Jitter (DJ): 0.000ns  
Phase Error (PE): 0.000ns

| Location                | Delay type      | Delay(ns)                                                            | Physical Resource                                                   |
|-------------------------|-----------------|----------------------------------------------------------------------|---------------------------------------------------------------------|
|                         |                 |                                                                      | Logical Resource(s)                                                 |
| SLICE_X76Y99.B0         | Tck0            | 0.467                                                                | Inst_FPA_RX/LVDS_SYS_I_d2                                           |
| SLICE_X80Y100.C5        | net (fanout=1)  | 0.564                                                                | Inst_FPA_RX/LVDS_SYS_I_d2                                           |
| SLICE_X80Y100.C         | Tilo            | 0.094                                                                | Inst_FPA_RX/Inst_FIFO/U0/xst_fifo_lut09749_0                        |
| RAMB36_X5Y17.WEALL1     | net (fanout=15) | 1.566                                                                | lut09749_0                                                          |
| RAMB36_X5Y17.CLKARDCLKL | Trock_WEA       | 0.624                                                                | Inst_FPA_RX/Inst_FIFO/U0/xst_fifo_Inst_FPA_RX/Inst_FIFO/U0/xst_fifo |
| Total                   |                 | 3.315ns (1.185ns logic, 2.130ns route)<br>(35.7% logic, 64.3% route) |                                                                     |

# 4. 载人航天FPGA软件测试验证



## ■ 时序分析

### 保持时间分析在最小工况下分析

```
TS_CLK_LVDS_I = PERIOD NET "CLK_LVDS_I_BUFGP/IBUFG" 16.667 ns HIGH 60%
 |- Setup paths
 |- 3.647 From Inst_FPA_RX/LVDS_SY...5_noinit.ram/SDP.SINGLE_PRIM18.SDP
 |- 3.647 From Inst_FPA_RX/LVDS_SY...5_noinit.ram/SDP.SINGLE_PRIM18.SDP
 |- 3.757 From Inst_FPA_RX/LVDS_SY...5_noinit.ram/SDP.SINGLE_PRIM18.SDP
 |- Hold paths
 |- 0.330 From Inst_FPA_RX/Inst_FI..._sync_fifo.g10.wr/wpnr/count_d1_2
 |- 0.338 From Inst_FPA_RX/Inst_FI..._sync_fifo.g10.wr/wpnr/count_d1_7
 |- 0.359 From Inst_FPA_RX/Inst_FI...5_noinit.ram/SDP.SINGLE_PRIM18.SDP
```

Slack (hold path): 0.330ns (requirement - (clock path skew + uncertainty - data path))  
Source: [Inst\\_FPA\\_RX/Inst\\_FIFO/U0/xst\\_fifo\\_generator/qconvfifo.rf/grf\\_rf/gntv\\_or\\_sync\\_fifo.q10.wr/wpnr/count\\_2 \(FF\)](#)  
Destination: [Inst\\_FPA\\_RX/Inst\\_FIFO/U0/xst\\_fifo\\_generator/qconvfifo.rf/grf\\_rf/gntv\\_or\\_sync\\_fifo.q10.wr/wpnr/count\\_d1\\_2 \(FF\)](#)  
Requirement: 0.000ns  
Data Path Delay: 0.316ns (Levels of Logic = 0)  
Clock Path Skew: -0.014ns (0.120 - 0.134)  
Source Clock: CLK\_LVDS\_I\_BUFGP rising at 16.667ns  
Destination Clock: CLK\_LVDS\_I\_BUFGP rising at 16.667ns  
Clock Uncertainty: 0.000ns

| Minimum Data Path: <a href="#">Inst_FPA_RX/Inst_FIFO/U0/xst_fifo_generator/qconvfifo.rf/grf_rf/gntv_or_sync_fifo.q10.wr/wpnr/count_2</a> to <a href="#">Inst_FPA_RX/Inst_FIFO/U0/xst_fifo...</a> |                |                                                                      |                                                                                                                                                                                                                                                             |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Location                                                                                                                                                                                         | Delay type     | Delay(ns)                                                            | Physical Resource                                                                                                                                                                                                                                           |
|                                                                                                                                                                                                  |                |                                                                      | Logical Resource(s)                                                                                                                                                                                                                                         |
| SLICE_X66Y91.CQ                                                                                                                                                                                  | Tck0           | 0.373                                                                | <a href="#">Inst_FPA_RX/Inst_FIFO/U0/xst_fifo_generator/qconvfifo.rf/grf_rf/gntv_or_sync_fifo.q10.wr/wpnr/count&lt;3&gt;</a><br><a href="#">Inst_FPA_RX/Inst_FIFO/U0/xst_fifo_generator/qconvfifo.rf/grf_rf/gntv_or_sync_fifo.q10.wr/wpnr/count_2</a>       |
| SLICE_X67Y91.DX                                                                                                                                                                                  | net (fanout=3) | 0.140                                                                | <a href="#">Inst_FPA_RX/Inst_FIFO/U0/xst_fifo_generator/qconvfifo.rf/grf_rf/gntv_or_sync_fifo.q10.wr/wpnr/count&lt;2&gt;</a>                                                                                                                                |
| SLICE_X67Y91.CLK                                                                                                                                                                                 | Tckdi (-Th)    | 0.197                                                                | <a href="#">Inst_FPA_RX/Inst_FIFO/U0/xst_fifo_generator/qconvfifo.rf/grf_rf/gntv_or_sync_fifo.q10.wr/wpnr/count_d1&lt;2&gt;</a><br><a href="#">Inst_FPA_RX/Inst_FIFO/U0/xst_fifo_generator/qconvfifo.rf/grf_rf/gntv_or_sync_fifo.q10.wr/wpnr/count_d1_2</a> |
| Total                                                                                                                                                                                            |                | 0.316ns (0.176ns logic, 0.140ns route)<br>(55.7% logic, 44.3% route) |                                                                                                                                                                                                                                                             |

## 4. 载人航天FPGA软件测试验证



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

### ■ 常见时序问题的解决方案

1、**建立时间违例**，建立时间发生违例表明设计中**存在传输延迟过大的路径**，导致该设计无法在既定频率和占空比的输入时钟情况下正常工作，通常有以下几种解决方法：

- 通过修改设计代码，降低寄存器之间的组合逻辑复杂程度；
- 修改综合约束，如设为速度优先；
- 在布局布线时重新进行时序约束；
- 设计降频使用。

## 4. 载人航天FPGA软件测试验证



### ■ 常见时序问题的解决方案

2、**保持时间违例**，保持时间发生违例表明设计中存在传输延迟过小的路径，导致该设计在任何输入时钟频率下均无法正常工作，通常有以下几种解决方法：

- 修改设计代码，增加寄存器之前的组合逻辑复杂程度；
- 修改设计代码，在相应寄存器间插入延时单元或 Buffer 并设置不被优化的约束；
- 修改设计代码，调整时钟树结构以减小 Skew，比如将时钟从专用时钟管脚输入，或将时钟显式使用 BUFG 资源“上树”；
- 在布局布线时重新进行时序约束。

## 4. 载人航天FPGA软件测试验证



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

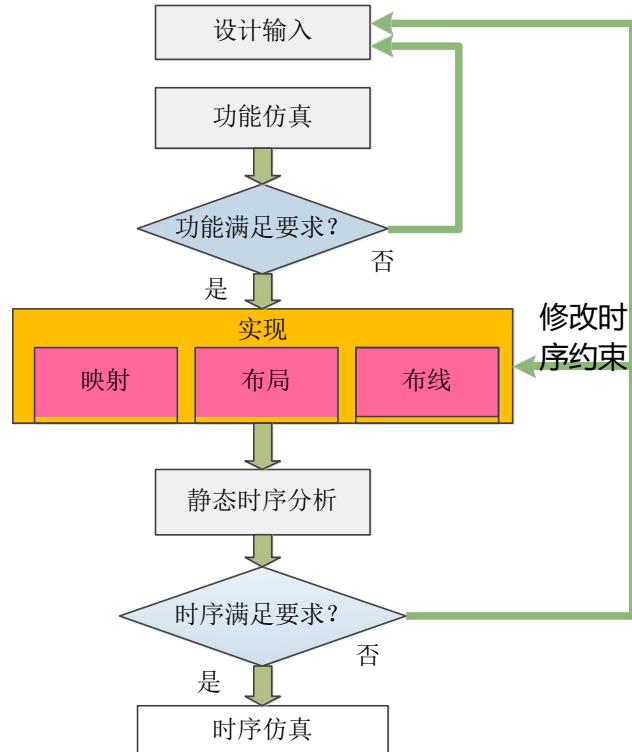
### (3) 时序仿真

## 4. 载人航天FPGA软件测试验证



- **功能仿真**: 又叫前仿真、代码仿真、逻辑仿真等，是在**RTL层**进行的仿真，其特点是不考虑构成电路的逻辑和门的时间延迟，着重考虑电路在理想环境下的行为和设计构想是否一致。
- **时序仿真**: 又叫后仿真，在电路已经映射到特定的工艺环境(器件)后，将电路的路径延时和门延时考虑进对电路行为的影响后，来比较电路的行为是否还能够在一定的条件下满足设计构想。时序仿真可以称为**动态时序分析**。

## 4. 载人航天FPGA软件测试验证



- **时序仿真在设计流程中的位置：在功能仿真和静态时序分析之后。**



## 4. 载人航天FPGA软件测试验证

### ■ 载人航天FPGA时序分析建议：

- **难点：**时序仿真流程与功能仿真验证相似，但是仿真**效率低**，用时序仿真方法达到功能仿真验证中的语句、分支、条件、表达式、状态机等100%覆盖较困难，且极为耗时，可能严重影响设计进度，尤其是对于**大规模、高复杂度**的FPGA产品；
- **建议：**用**功能仿真**验证设计功能，达到100%覆盖率；时序仿真只运行**典型功能 case**，覆盖**三种工况**；更全面的时序验证采用**静态时序分析**的方法；

## 4. 载人航天FPGA软件测试验证



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

### ■ 载人航天FPGA软件研制技术要求

- 应进行时序仿真和功能仿真结果的一致性比较，确保布局布线后的门级网表和 RTL 代码逻辑等价性，避免工具引入错误，也可采用**逻辑等价性检查工具**进行 RTL 代码与门级网表的逻辑等价性检查。

## 4. 载人航天FPGA软件测试验证



### ■ 逻辑等价性检查工具

| 序号 | 公司       | 软件名称      | 适用平台 |
|----|----------|-----------|------|
| 1  | Synopsys | Formality | UNIX |
| 2  | Mentor   | FormalPro | UNIX |
| 3  | Cadence  | Conformal | UNIX |

## 4. 载人航天FPGA软件测试验证



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study,UCAS

### (4) 配置项测试

## 4. 载人航天FPGA软件测试验证



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

- 配置项编程下载，在真实目标机上或模拟环境中运行，外围可以是实物、半实物和配置项仿真接口，确认软件是否达到了软件需求规格说明所规定的各项要求

## 4. 载人航天FPGA软件测试验证



國科大杭州高茅研究院  
Hangzhou Institute for Advanced Study, UCAS

分系统验证测试  
系统验证测试  
研制总结



# THANKS



國科大杭州高等研究院  
Hangzhou Institute for Advanced Study, UCAS