本历史

文档更新记录	文档名:	Lab02_时序器件行为与数字电路设计调试
	版本号	V0.1
	创建人:	计算机体系结构研讨课教学组
	创建日期 :	2019-09-02

更新历史

序号	更新日期	更新人	版本号	更新内容
1	2019/09/02	宋玥坤	-	草稿。
2	2017/09/04	邢金璋	V0.1	初版。

文档信息反馈: xingjinzhang@loongson.cn

1 实验二 时序器件行为与数字电路设计调试

在学习并尝试本章节前，你需要具有以下环境和能力：

(1) 装有 Vivado 的电脑一台。

(2) 熟悉 Vivado，并能初步使用。

如果对 Vivado 不熟悉，请参考课程讲义中的第一讲内容。

(3) 初步掌握 Verilog 的简单语法。

(4) 熟悉龙芯体系结构实验箱（Artix-7）。

通过本章节的学习，你将获得：

(1) Vivado 生成 RAM IP 的方法。

(2) 对同步、异步 RAM 深入的理解。

(3) 通过仿真波形描述设计行为的能力。

(4) 对设计进行综合并获取时序报告和资源报告的能力。

(5) 对数字电路设计出错类型的识别、调试的能力。

1.1 实验目的

1. 了解寄存器堆的原理。
2. 了解同步 RAM 和异步 RAM 的原理。
3. 理解同步 RAM 和异步 RAM 的区别。
4. 掌握调用 Xilinx 库 IP 实例化 RAM 的设计方法。
5. 学会识别常见波形异常，理解其产生原因，并能修正。
6. 熟悉并运用 verilog 语言进行电路设计。

1.2 实验设备

1. 装有 Xilinx Vivado 的计算机一台。
2. 龙芯体系结构教学实验箱（Artix-7）一套。

1.3 实验任务

本次实验分为三个子任务：

1. 子任务一：寄存器堆仿真。建立寄存器堆工程，加入我们提供的寄存器堆的源码和仿真文件，建立工程，并进行仿真，得到正确的波形，根据波形描述寄存器堆的读写行为。
2. 子任务二：同步、异步 RAM 仿真、综合实现。建立同步 RAM 工程，调用 Xilinx 库 IP 实例化一块深度为 65536、宽度为 32 的同步 RAM（Block Memory），加入我们提供的仿真文件和约束文件。另外，建立一个异步 RAM 工程，调用 Xilinx 库 IP 实例化一块深度为 65536、宽度为 32 的异步 RAM（Distributed Memory），加入我们提供的仿真文件和约束文件。对两个工程分别进行仿真，根据波形描述同步、异步 RAM 的读写行为和异同。对两个工程分别进行综合（synthesis）和实现（implementation），查看同步、

异步 RAM 的时序和资源利用率，分析其异同。

3. 子任务三：数字电路设计调试。利用我们提供的数字电路设计（show_sw）的源码、仿真文件和约束文件，建立一个工程，对设计源码进行调试，找出其中的所有 bug（共 5 处），记录下所有 bug 的调试过程和机理分析，要求最后的设计能正确的运行仿真和上板验证。
4. 本次实验要求每个人独立提交实验报告，以报告评分和现场检查评分作为最后的实验得分：
   (1) 报告评分：子任务一、二要求有相应的截图和分析，子任务三要求记录下所有 bug 的调试过程和机理分析。
   (2) 现场检查评分：子任务一的检查包含仿真检查；子任务二的检查包含仿真检查、综合实现检查；子任务三的检查包含仿真检查和上板检查。

1.4 实验检查

检查前需提交实验报告（每人一份）。本次实验在 2019 年 9 月 10 日进行检查。

现场分仿真检查和上板检查：

1) 仿真检查：子任务一、二对照波形进行描述读写行为。
2) 上板检查：子任务三查看上板行为。

现场检查要求能正确应对检查者的提问，并根据要求进行正确的操作演示

1.5 实验提交

提交的作品包括纸质档和电子档。

(1) 纸质档提交

提交方式：课上现场提交，每人都必须要有。

截止时间：2019 年 9 月 10 日 18:10。

提交内容：纸质档 lab2 实验报告。

(2) 电子档提交

提交方式：打包上传到 Sep 课程网站 lab2 作业下，每人都必须要有。

截止时间：2019 年 9 月 10 日 18:10（具体以课程通知为准）。

提交内容：电子档 lab2 实验报告，为 pdf 文档，文件名是“lab2_箱子号_学号”（请将其中的“箱子号”替换为本组箱子号，“学号”替换为自己的学号）。

1.6 实验环境

本次实验只提供工程需要使用到的源码文件、仿真文件和约束文件。需要同学们自行新建工程，添加这三类文件进行实验。

提醒：子任务二需要新建两个工程，一个同步 RAM 的工程，一个异步 RAM 的工程，但这两个工程使用的约束文件是相同的（ram.xdc），使用的仿真文件也是相同的（ram_tb.v）。

|--regfile_task/	目录，包含子任务一（寄存器堆仿真）的源码文件和仿真文件。
| |--regfile.v	寄存器堆源码文件。
| |--rf_tb.v	寄存器堆仿真文件。
|	
|--ram_task/	目录，包含子任务二（同步、异步 RAM）的源码文件、约束文件和仿真文件。

|	|--block_ram_top.v	同步 RAM（Block RAM）的源码顶层文件。
|	|--distributed_ram_top.v	异步 RAM（Distributed RAM）的源码顶层文件。
|	|--ram.xdc	两种 RAM 的仿真约束文件，用于综合和实现。
|	|--ram_tb.v	两种 RAM 的仿真文件，用于仿真。
|		
|--debug_task/		目录，包含子任务三（数字电路设计调试）的源码文件、约束文件和仿真文件。
|	|--show_sw.v	数字电路设计的源码文件。
|	|--show_sw.xdc	数字电路设计的约束文件，用于综合和实现。
|	|--tb.v	数字电路设计的仿真文件，用于仿真。

1.7 子任务一：寄存器堆仿真

1.7.1 寄存器堆接口说明

本实验提供的寄存器堆为单发射 MIPS CPU 中需要使用到两读一写的结构，也就是有两个读端口（读端口没有使能位控制，表示永远使能）、一个写端口。接口信号如下表：

表 1-1 寄存器堆接口信号列表

名称	宽度	方向	描述
时钟：			
clk	1	input	时钟信号
读端口一：			
raddr1	5	input	寄存器堆读地址 1。
rdata1	32	output	寄存器堆读返回数据 1。
读端口二：			
raddr2	5	input	寄存器堆读地址 2。
rdata2	32	output	寄存器堆读返回数据 2。
写端口：			
we	1	input	寄存器堆写使能
waddr	5	input	寄存器堆写地址
wdata	32	input	寄存器堆写数据

1.7.2 寄存器堆实验步骤

请参考下列步骤进行实验：

(1) 使用 Vivado 新建一个工程。

(2) 点击“Add Sources”，选择添加设计源码（design sources），加入 lab2 任务包里的 regfile_task/regfile.v。

(3) 再次点击“Add Sources”，选择添加仿真源码（simulation sources），加入 lab2 任务包里的 regfile_task/rf_tb.v。

(4) 对工程进行仿真测试，结合波形观察寄存器堆的读写行为。

(5) 对波形的表现进行理解思考，记录到实验报告中。

1.7.3 实验报告提醒

请在撰写实验报告时注意以下几点：

(1) 实验报告要求描述寄存器堆的读写行为，应该是先给出一个波形截图，对照该截图进行描述。

(2) 波形截图应当信号分明（请合理使用分割（Divider）、分组（Group）和颜色变化来使信号分明），请在仿真波形中添加标志线（Marker），在对该截图描述时，采用类似“第一个 Marker 时……，到第二个 Marker 时……”的方式描述。

1.8 子任务二：同步、异步 RAM 仿真、综合实现

1.8.1 RAM 顶层接口说明

本实验要求对同步、异步 RAM 各自建立一个工程，调用 Xilinx 库 IP 实例化同步、异步 RAM，但是都会提供一个设计的顶层文件，将他们封装成相同的模块名和接口。封装后的 RAM 接口信号如下表：

表 1-2 RAM 顶层接口信号列表

名称	宽度	方向	描述
clk	1	Input	时钟信号
ram_wen	1	Input	RAM 的写使能信号：为 1 表示写入操作；为 0 表示读取操作。
Ram_addr	16	Input	RAM 的地址信号，读和写的地址都由该信号指示。
Ram_wdata	32	Input	RAM 的写数据信号，表示写入的数据。
ram_rdata	32	Output	RAM 的读数据信号，表示读出的数据。

注意：封装后的 RAM 接口没有“使能”（或者称为“片选”）信号，表示永远使能（“使能”信号在 RAM 内部恒为 1）。

1.8.2 RAM 实验步骤

RAM 实验需要新建两个工程，一个同步 RAM 工程，一个异步 RAM 工程。

新建同步 RAM 工程的参考步骤如下：

(1) 使用 Vivado 新建一个工程。

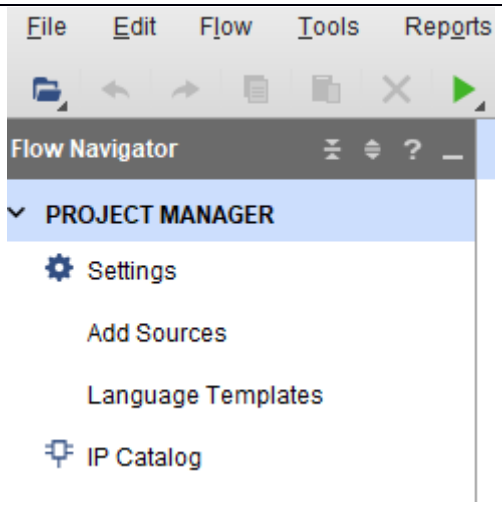
(2) 点击“Add Sources”，选择添加设计源码（design sources），加入 lab2 任务包里的 ram_task/block_ram_top.v。

(3) 再次点击“Add Sources”，选择添加约束文件（constraints），加入 lab2 任务包里的 ram_task/ram.xdc。

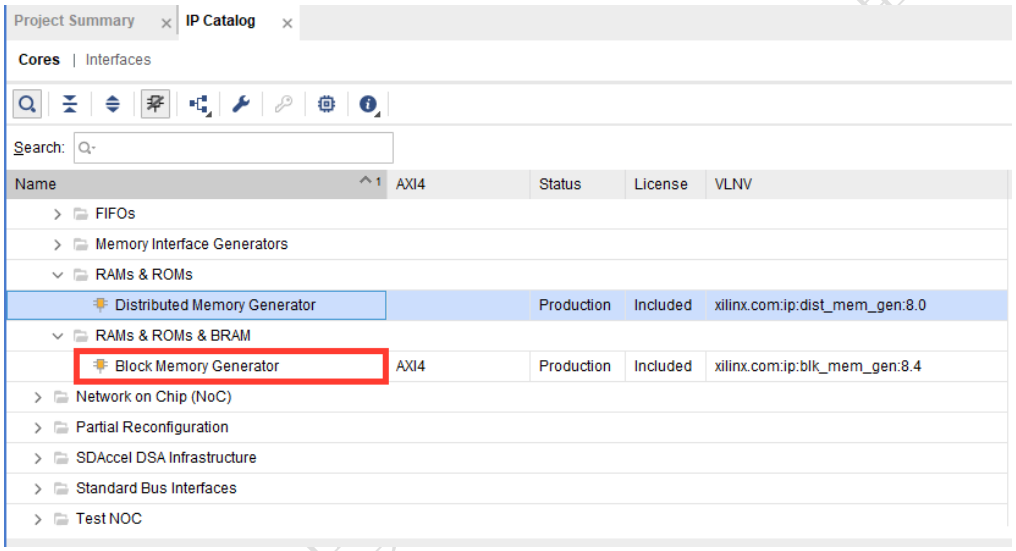
(4) 再次点击“Add Sources”，选择添加仿真源码（simulation sources），加入 lab2 任务包里的 ram_task/ram_tb.v。

(5) 调用 Xilinx 库 IP 生成 Block RAM（深度为 65536，宽度为 32，使能信号为一直使能）：

a) 在“Project manager”里点击“IP Catalog”

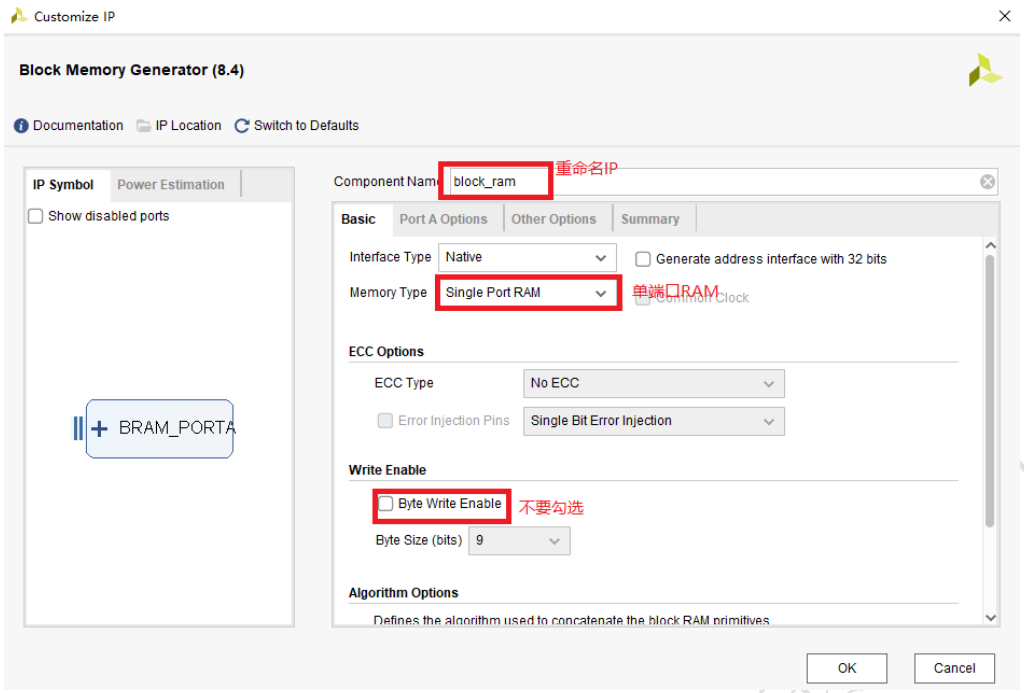


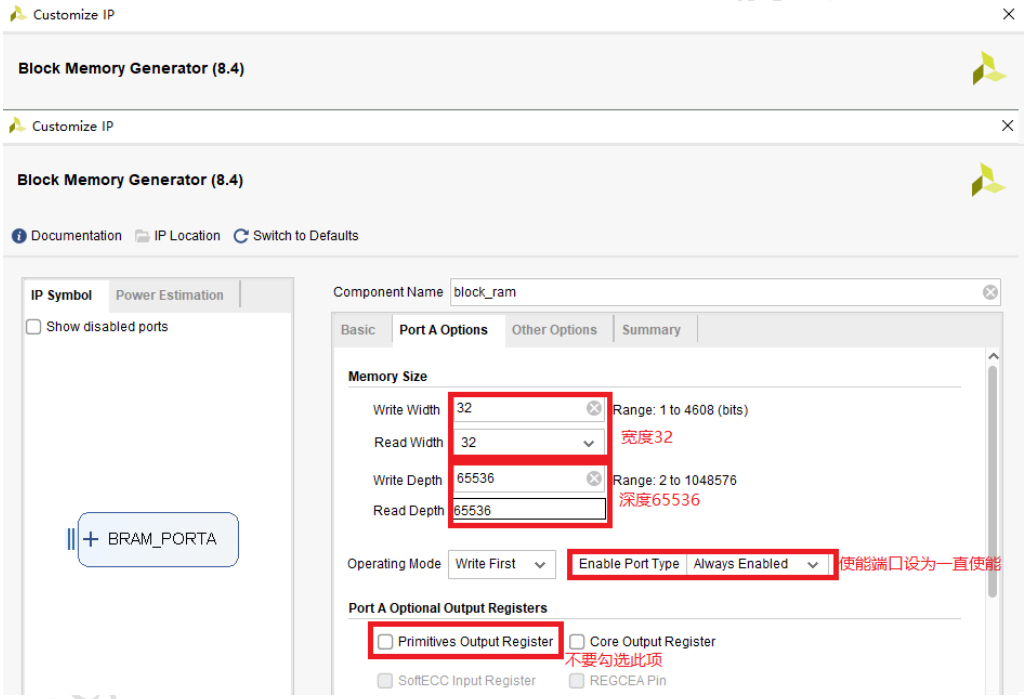
b) 在右侧列表中双击选择”Memories and Storage Elements”->”RAMs & ROMs & BRAM”中的”Block Memory Generator”:



c) 在打开的 IP 定制界面中设置 RAM 参数：

在“Basic”界面将 IP 重命名为 block_ram，设为单端口 RAM，“Byte write Enable”不要勾选；在“Port A Options”界面将 RAM 深度设为 65536、宽度设为 32，使能端口设为“Always Enabled”，不要勾选“Primitives Output Register”。其他保持默认即可，点击“OK”即可。





新建异步 RAM 工程的参考步骤如下：

(1) 使用 Vivado 新建一个工程。

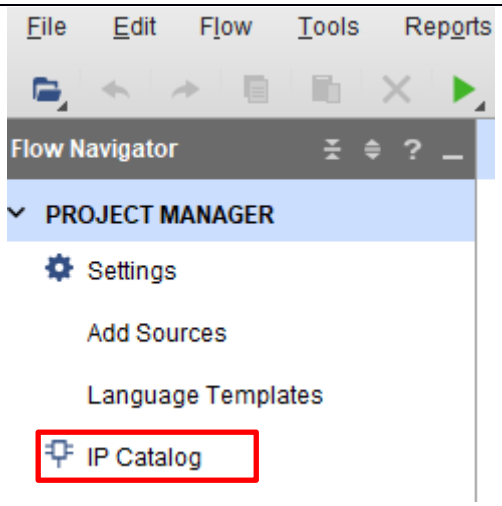
(2) 点击“Add Sources”，选择添加设计源码（design sources），加入 lab2 任务包里的 ram_task/distributed_ram_top.v。

(3) 再次点击“Add Sources”，选择添加约束文件（constraints），加入 lab2 任务包里的 ram_task/ram.xdc。

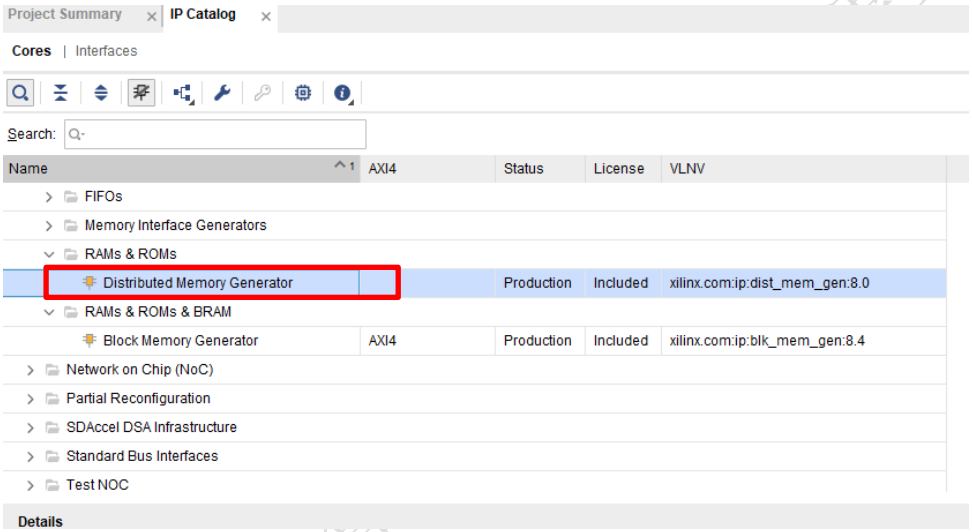
(4) 再次点击“Add Sources”，选择添加仿真源码（simulation sources），加入 lab2 任务包里的 ram_task/ram_tb.v。

(5) 调用 Xilinx 库 IP 生成 Distributed RAM（深度为 65536，宽度为 32）：

a) 在“Project manager”里点击“IP Catalog”

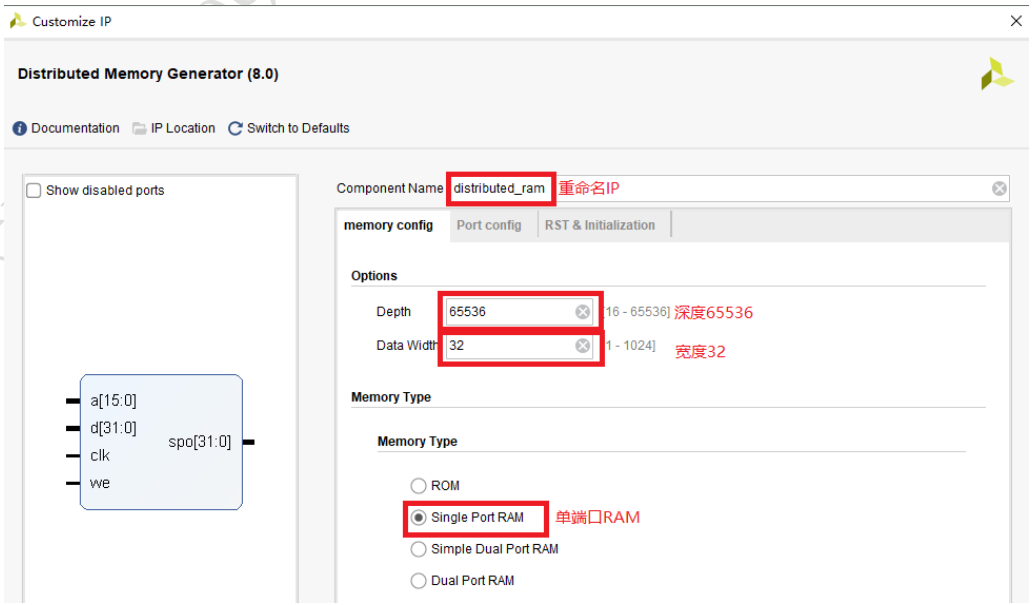


b) 在右侧列表中双击选择”Memories and Storage Elements”->”RAMs & ROMs “中的”Distributed Memory Generator”:



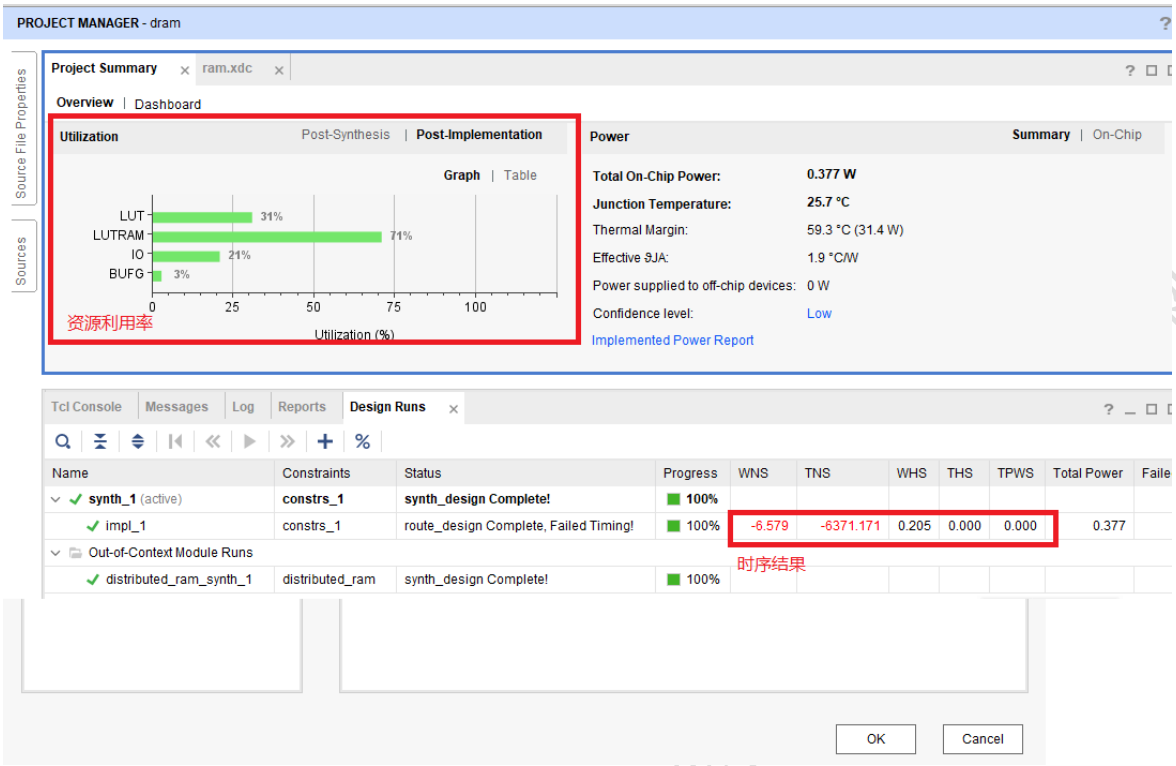
c) 在打开的 IP 定制界面中设置 RAM 参数：

在“memory config”界面将 IP 重命名为 distributed_ram，设为单端口 RAM，深度设为 65536、宽度设为 32。其他保持默认即可，点击“OK”即可。



在完成工程的创建后，需要对它们进行仿真，对比他们的读写行为。

在完成仿真后，需要对它们进行综合（synthesis）和实现（implementation），查看时序结果和资源利用率，如下图，并结合读写时序进行分析。



总结以上分析，对比同步、异步 RAM 的优劣。

1.8.3 实验报告提醒

请在撰写实验报告时注意以下几点：

(1) 实验报告要求描述同步、异步 RAM 的读写行为，应当和寄存器堆的读写行为分析一样，以图文结合的方式进行描述，截图中使用标志线（Marker）来辅助描述。

(2) 对比分析两种 RAM 的读写行为，并与寄存器堆的读写行为进行对比分析。

(3) 结合各自的时序和资源报告，描述两者的差异。请在报告中加入时序和资源报告的截图。

(4) 根据两者的差异，谈一谈两种 RAM 在生成大块 RAM(实验中所用大小)时的优劣。

1.8.4 RAM 实验中特别注意的点

在实验过程中请特别注意以下几点：

(1) 在生成 IP 时，请将对应 IP 命名为 block_ram 和 distributed_ram，如错误命名 ip 将会报错。若遇到已生成完 IP 无法改名，删除该 IP，重新生成。

(2) 生成 IP 时，可以点击窗口左侧的图查看接口信息。当参数正确时，端口名和宽度应与实验指定的顶层文件中调用相对应。

(3) 有兴趣的同学可以自行勾选/勾掉 RAM 设置中的 Registerd/Non Registered 选项，并根据仿真波形对比该选项的作用。

(4) 对程序进行综合之前请确保已正确加载约束文件(ram.xdc)。

(5) 添加 testbench 时请注意选择 add simulation source，否则会导致顶层文件错误，综合结果不正确。

(6) 对程序进行综合时根据所用计算机不同，综合时间会有一定的差异。综合时可能会耗费大量时间，请提前计划，安排好时间。

(7) 时序报告和资源报告的生成需要查看综合(synthesis)并且实现(implementation)完成后的结果，可以在左侧

导航栏→实现结果(Open Implementation design)→生成时序报告(report timing summary)或生成资源报告(report utilization)。

1.9 子任务三：数字电路设计调试

1.9.1 数字电路设计功能说明

本实验提供的是一个有 5 个 bugs 的数字电路设计源码。该设计的正确功能是：

1) 获取开发板最右侧 4 个拨码开关的状态（记为“拨上为 1，拨下为 0”，实际开发板上拨码开关的电平是“拨上为低电平，拨下为高电平”），共有 16 个状态（数字编号是 0~15）。
2) 最左侧数码管实时显示 4 个拨码开关的状态。数码管只支持显示 0~9，如果拨码开关状态是 10~15，则数码管的显示状态不更改（显示上一次的显示值）。
3) 最右侧的 4 个单色 LED 灯会显示上一次的拨码开关的状态，支持显示 0~15（拨码开关拨上，对应 LED 灯亮）。
4) 比如：初始状态，4 个拨码开关拨下，按复位键，则数码管显示 0，LED 灯都不亮；拨码开关拨为 1，则数码管显示 1，LED 灯还是都不亮；拨码开关再拨为 3，则数码管显示 3，LED 灯显示 1。

提供的设计源码中包含 5 个 bugs，其中 4 个是讲义二中提到的波形异常里的前 4 中情况：波形为“Z”，波形为“X”，波形停止和越沿采用；另外的 1 个 bug 是功能 bug。

1.9.2 顶层接口说明

本实验提供的示例设计的顶层接口如下：

表 1-3 示例设计的顶层信号列表

名称	宽度	方向	描述
Clk	1	input	时钟信号
Resetn	1	input	复位信号
Switch	4	input	对应开发板上最右侧四个拨码开关
num_csn	8	output	数码管的片选信号
num_a_g	7	output	数码管的 7 段信号
Led	4	output	对应开发板上最右侧 4 个单色 LED 灯。

1.9.3 Debug 实验步骤

请参考下列步骤进行实验：

(1) 使用 Vivado 新建一个工程。

(2) 点击“Add Sources”，选择添加设计源码（design sources），加入 lab2 任务包里的 debug_task/show_sw.v。

(3) 再次点击“Add Sources”，选择添加约束文件（constraints），加入 lab2 任务包里的 debug_task/show_sw.xdc。

(4) 再次点击“Add Sources”，选择添加仿真源码（simulation sources），加入 lab2 任务包里的 debug_task/tb.v。

(5) 理解示例设计的功能，分析仿真顶层 tb.v，理解仿真的行为。

注意开发板上拨码开关的电平是“拨上为低电平，拨下为高电平”，单色 LED 灯的电平行为是“高电平不亮，低电平亮”。仿真顶层 tb.v 也是按此电平设计的。

(6) 进行仿真，并充分利用仿真的辅助小技巧（分割（Divider）、分组（Group）、颜色变化、标志线（Marker）等等）进行调试，找出所有的 bug。在调试的过程中，注意记录调试过程和机理分析的过程，以作为实验报告。

(7) 仿真完成后，进行综合（synthesis）、实现（implementation）并生成 bit 流文件（generate bitstream）。

(8) Bit 流文件生成好后，连接开发板，进行上板验证。

(9) 根据记录的调试过程和 bug 机理分析的过程，完成实验报告的撰写。

1.9.4 实验报告提醒

请在撰写实验报告时注意以下一点：

(1) 重点记录调试过程和 bug 机理分析，要求图文结合的方式进行藐视。