实验五 单周期 MIPS 处理器的设计与实现

一． 实验目的

1. 熟悉 MIPS 处理器的常用指令集（10 条）
2. 掌握单周期处理器数据通路和控制单元的设计方法
3. 基于增量方式，实现单周期 MIPS 处理器；
4. 基于测试用例对所设计的单周期 MIPS 处理器进行功能验证。

二． 实验环境

1. 操作系统：Windows 10 或 Ubuntu 16.04
2. 开发环境：Xilinx Vivado 2018.2
3. 硬件平台：远程 FPGA 硬件云平台

三． 实验原理

处理器（CPU）本质上是一个复杂的数字时序电路。通常，时序电路由记忆部件（如寄存器、存储器等）和组合逻辑构成。记忆部件用于保存电路的工作状态，而组合逻辑则由逻辑门组成，提供电路的所有逻辑功能。在组合逻辑的作用下，电路从一个状态转化为另一个状态，这样的电路也称为“状态机”。因此，单周期 MIPS 处理器在概念上也可以被看作一个大规模的状态机，如图 5-1 所示。其中，组合逻辑依据当前记忆部件中的值（即电路的现态）对指令进行处理，这个处理过程将再次修改记忆部件的值，使电路达到新状态（即电路的次态）。

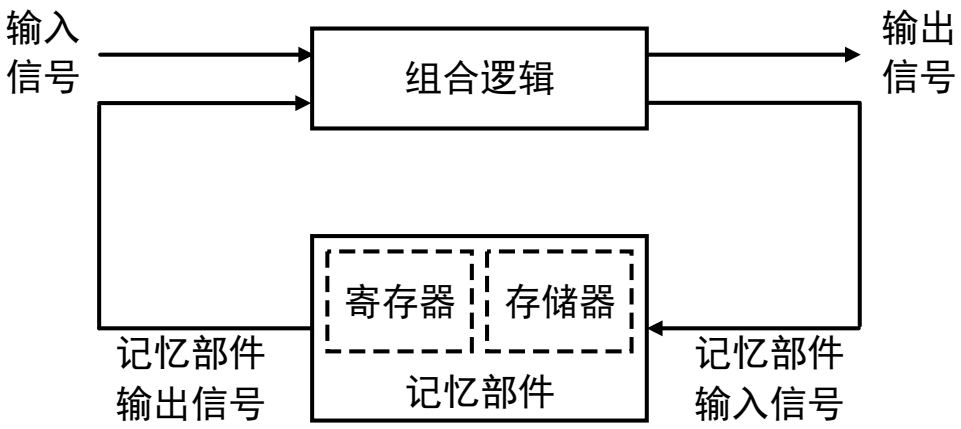


图 5-1 处理器的概念模型（状态机）

因此，在设计单周期 CPU 这样复杂的时序电路系统时，通常的方法是从包含记忆部件的硬件开始。这些元件包括**存储器**和**寄存器**，寄存器又可分为**程序计数器**和**寄存器文件**。然后，在这些存储元件之间增加组合逻辑基于当前状态计算新的状态。从指令存储器中读取指令，然后译码时访问寄存器文件获取操作数，再使用加载和存储指令从数据存储器中读取和写入数据。图 5-2 给出了具有 4 种状态元件（**程序计数器 PC**、**寄存器文件**、**指令存储器**和**数据存储器**）的框图。

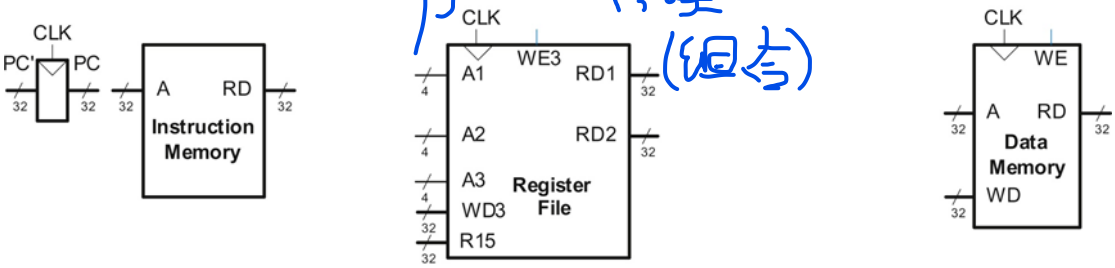


图 5-2 单周期 MIPS CPU 的状态元件

程序计数器 PC 是一个普通的 32 位寄存器。它的输出 PC 指向当前指令，它的输入 PC'表示下一条指令的地址。

指令存储器采用异步读的 ROM（读操作不受时钟控制）实现，具有一个读端口，包括 32 位指令地址输入 A，然后从这个地址读 32 位指令并传送到读数据输出 RD 上。

寄存器文件（也称为寄存器堆）由 **32 个 32 位寄存器**组成。具有有**两个读端口**和**一个写端口**。读端口具有 5 位地址输入 A1 和 A2 每个用于指定 32 个寄存器中的一个作为源操作数。它们可以读 32 位寄存器的值并分别传送到 RD1 和 RD2 上。写端口具有 5 位地址 A3，32 位数据输入 WD，写入使能 WE3 和时钟信号 CLK。如果写使能为 1，则寄存器文件将在时钟上升沿将数据写入指定的寄存器中。寄存器文件采用**异步读/同步写**工作模式，即读操作不受时钟控制，而写操作必须由时钟进行同步。

数据存储器有**一个读端口**和**一个写端口**。如果写使能 WE 为 1，则在时钟上上沿将数据 WD 写入地址 A。如果写使能为 0，则从地址 A 将数据读到 RD。数据存储器也采用**异步读/同步写**工作模式。

对单周期 MIPS CPU 进行设计时，可将其体系结构分为两个部分分别设计，即**数据通路**和**控制单元**。数据通路对指令和数据进行操作，它包含存储器、寄存器、ALU 和复用器等部件。MIPS 是一个 32 位指令集体系结构，因此可以使用 32 位数据通路。控制单元从数据通路中接收当前指令，并控制数据通路如何执行这条指令。具体而言，控制单元产生复用器、寄存器使能和存储器写使能信号来控制数据通路的操作。

单周期 MIPS CPU 具体设计细节，即如何根据指令设计其数据通路和控制单元，请大家仔细阅读教材 7.3.1~7.3.3，并以此为参照完整本实验的设计要求。

四． 实验内容

基于 SystemVerilog HDL 设计并实现单周期 MIPS 处理器——MiniMIPS32。该处理器具有如下特点：

- 32 位数据通路

- 小端模式
- 支持 10 条指令：lw、sw、lui、ori、addiu、addu、slt、beq、bne 和 j
- 寄存器文件由 32 个 32 位寄存器组成，采用异步读/同步写工作模式
- 采用哈佛结构(即分离的指令存储器和数据存储器)，指令存储器由 ROM 构成，采用异步读工作模式；数据存储器由 RAM 构成，采用异步读/同步写工作模式

本实验的顶层模块 MiniMIPS32_SYS 如图 5-3 所示。该顶层模块由 4 部分构成，每部分的功能如下：

(1) MiniMIPS32 是本实验所需要设计完成的单周期 MIPS 处理器。输入端口 inst 和 dout 分别用于从指令存储器和数据存储器中接收指令和数据。输出端口 iaddr 和 daddr 用于传输访问指令存储器和数据存储器的地址。输出端口 we 用于给出数据存储器的写使能。输出端口 din 用于给出待写入数据存储器的数据。

(2) inst_rom 是指令存储器，用于存放所需要执行的指令。inst_rom 指令存储器的深度为 256，宽度为 32 位，故其容量为 1KB。inst_rom 具有一个输入端口 a 和输出端口 spo，其中输入端口 a 用于接收待访问指令的地址，其宽度为 8 位，输出端口 spo 用于输出指令，其宽度为 32 位。

(3) data_ram 是数据存储器，用于存放需要访问的数据。data_ram 数据存储器的深度为 256，宽度为 32 位，故其容量为 1KB。输入端口 daddr 为 8 位，用于给出访存地址。输入端口 din 为 32 位，用于给出待写入数据存储器中的数据。输入端口 we 为写使能信号，当其值“1”时，表示进行写操作，否则进行读操作。输出端口 spo 用于从数据存储器中读出数据。

(4) testled 是测试模块，用于判断当前执行的程序是否正确。如果正确，则

两个 LED 灯（led_r 和 led_g）分别为红色和不点亮状态，否则两个 LED 灯（led_r 和 led_g）分别为不点亮状态和绿色。

注意：本实验中同学们只需要实现单周期 MIPS 处理器模块，即 MiniMIPS32（图 5-3 中黄色模块），剩下的其余模块均已实现，由实验工程直接提供。

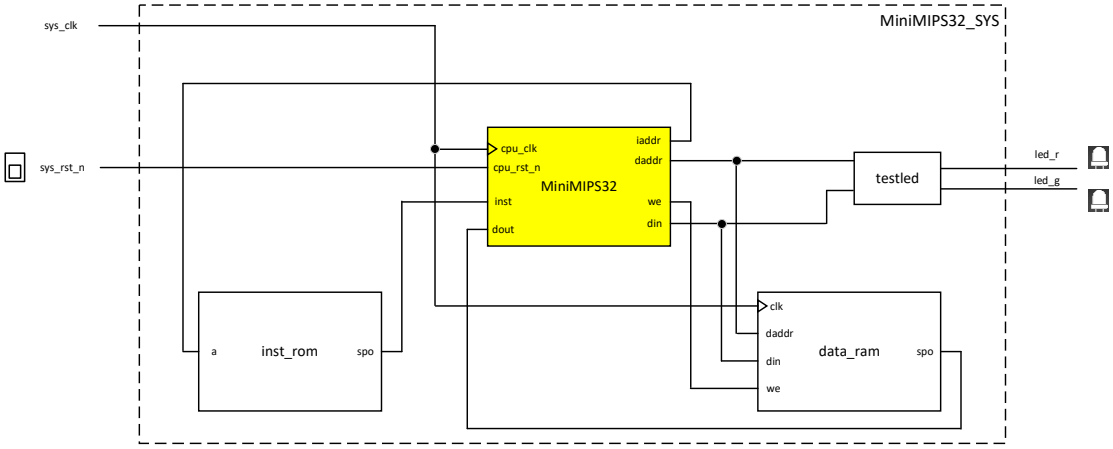


图 5-3 顶层模块 MiniMIPS32_SYS

表 5-1 给出了顶层模块 MiniMIPS32_SYS 的输入/输出端口。

表 5-1 输入/输出端口

端口名	方向	宽度（位）	作用
sys_clk	输入	1	系统输入时钟，主频为 25MHz（40ns）
sys_rst_n	输入	1	系统复位，低电平有效，连接拨动开关 S
led_r	输出	1	用于显示程序是否正确，红色 LED 灯。如果程序正确，不点亮，否则显示红色。
led_g	输出	1	用于显示程序是否正确，绿色 LED 灯。如果程序正确，显示绿色，否则不点亮。

参考教材中的 7.3.1~7.3.3 完成单周期 MIPS 处理器（MiniMIPS32）的设计（注意：**MiniMIPS32 采用小端模式**，而教材中采用的是大端模式，在本设计中需要针对小端模式的特点进行设计）

MiniMIPS32 处理器所支持的的 **10 条指令的功能描述以及指令编码，请参考教材的附录 B：MIPS 指令。**

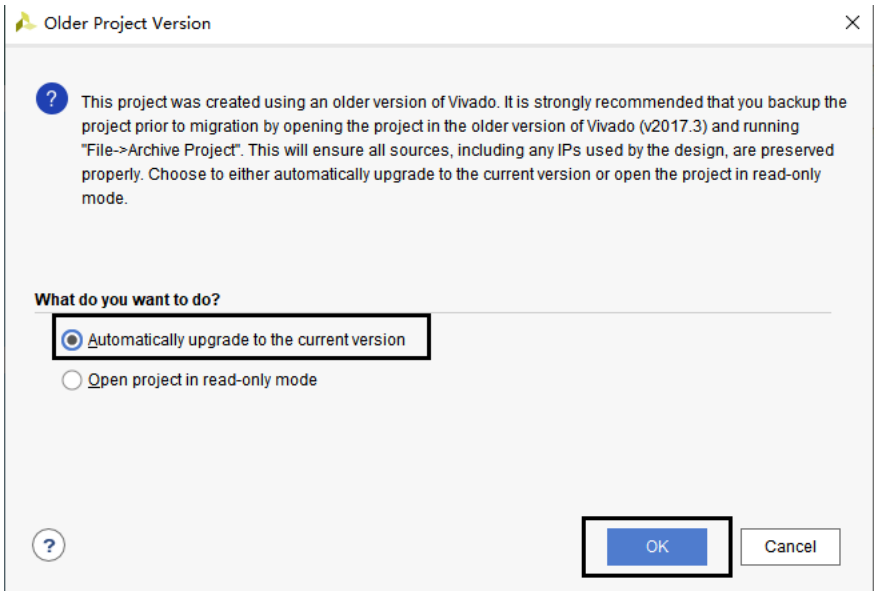
最终设计实现的单周期MIPS处理器能够运行所提供的6个测试用例**mem.S，i-type.S，r-type.S，branch.S，sort_sim.S 和 sort_board.S**。其中，前5个只能用于功能仿真；最后一个可以上传到远程FPGA硬件云平台完成功能验证，如果测试通过则LED灯led_g被点亮为绿色，否则LED灯led_r被点亮为红色。

五． 实验步骤

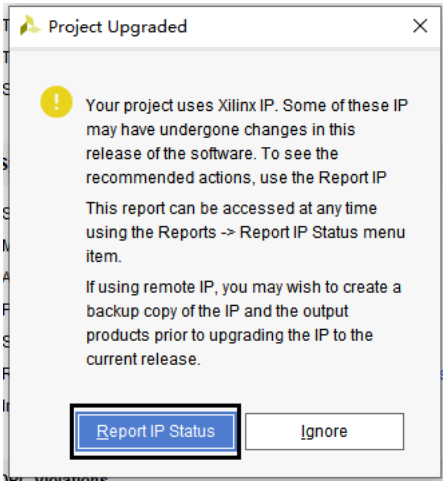
整个实验采用增量方式完成，每次只实现 2~3 条指令，仿真通过后再继续添加，直至支持所有 10 条指令，即边设计边测试的方式。切记不要实现全部指令后再进行测试!!!

步骤 1：打开工程并更新 IP

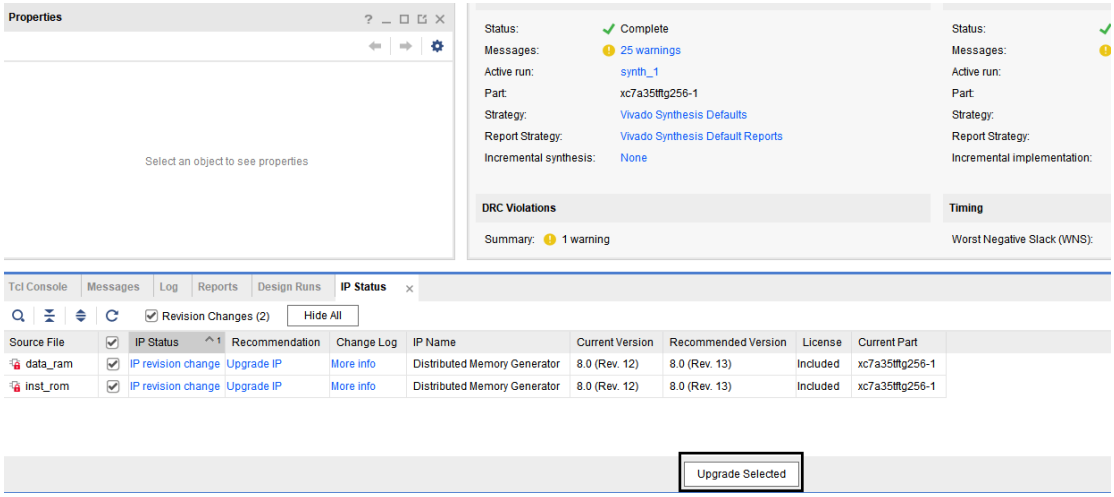
打开教师提供的实验工程 MiniMIPS32，若没有弹出如图 5-4(a)所示窗口，则直接跳到步骤 2。否则，选择“Automatically upgrate to the current version”，单击 OK 按钮。然后，在弹出的窗口中单击“Report IP Status”按钮，如图 5-4(b)所示。然后，在 Vivado 主窗口下方的 IP Status 标签中，单击“Update Selected”按钮，如图 5-4(c)所示。在弹出的窗口中，点击 OK 按钮，如图 5-4(d)所示。最后，如图 5-4(e)所示，在弹出的窗口中选择“Convert IP to Core Container and Set as Default in Project”，再单击 OK 按钮完成 IP 的更新。


Older Project Version
This project was created using an older version of Vivado. It is strongly recommended that you backup the project prior to migration by opening the project in the older version of Vivado (v2017.3) and running "File->Archive Project". This will ensure all sources, including any IPs used by the design, are preserved properly. Choose to either automatically upgrade to the current version or open the project in read-only mode.
What do you want to do?
Automatically upgrade to the current version
Open project in read-only mode
OK
Cancel

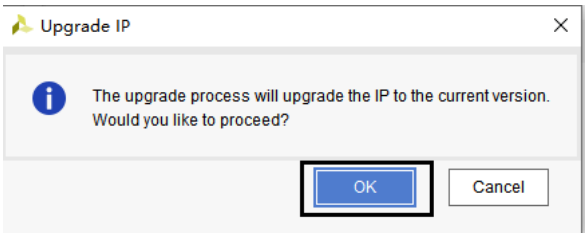
(a)


Project Upgraded
Your project uses Xilinx IP. Some of these IP may have undergone changes in this release of the software. To see the recommended actions, use the Report IP
This report can be accessed at any time using the Reports -> Report IP Status menu item.
If using remote IP, you may wish to create a backup copy of the IP and the output products prior to upgrading the IP to the current release.
Report IP Status
Ignore

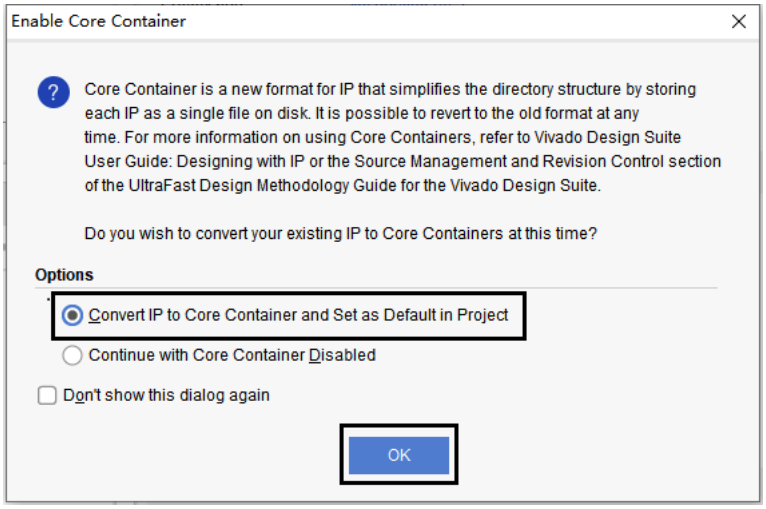
(b)


Properties
Select an object to see properties
Status: Complete
Messages: 25 warnings
Active run: synth_1
Part: xc7a35tftg256-1
Strategy: Vivado Synthesis Defaults
Report Strategy: Vivado Synthesis Default Reports
Incremental synthesis: None
DRC Violations
Summary: 1 warning
Timing
Worst Negative Slack (WNS):
Tcl Console
Messages
Log
Reports
Design Runs
IP Status
Revision Changes (2)
Hide All
Source File
IP Status
Recommendation
Change Log
IP Name
Current Version
Recommended Version
License
Current Part
data_ram
IP revision change
Upgrade IP
More info
Distributed Memory Generator
8.0 (Rev. 12)
8.0 (Rev. 13)
Included
xc7a35tftg256-1
inst_rom
IP revision change
Upgrade IP
More info
Distributed Memory Generator
8.0 (Rev. 12)
8.0 (Rev. 13)
Included
xc7a35tftg256-1
Upgrade Selected

(c)



(d)



(e)

图 5-4 打开工程并更新 IP

步骤 2：设计

(1). 如图 5-5 所示，目前工程中已经提供了完整的指令存储器 inst_rom、数据存储器 data_ram、顶层模块 MiniMIPS32_SYS.sv 和测试程序 MiniMIPS32_SYS_tb.sv。同学们只需设计实现支持 10 条指令的单周期 MIPS 处理器，即 MiniMIPS32.sv。

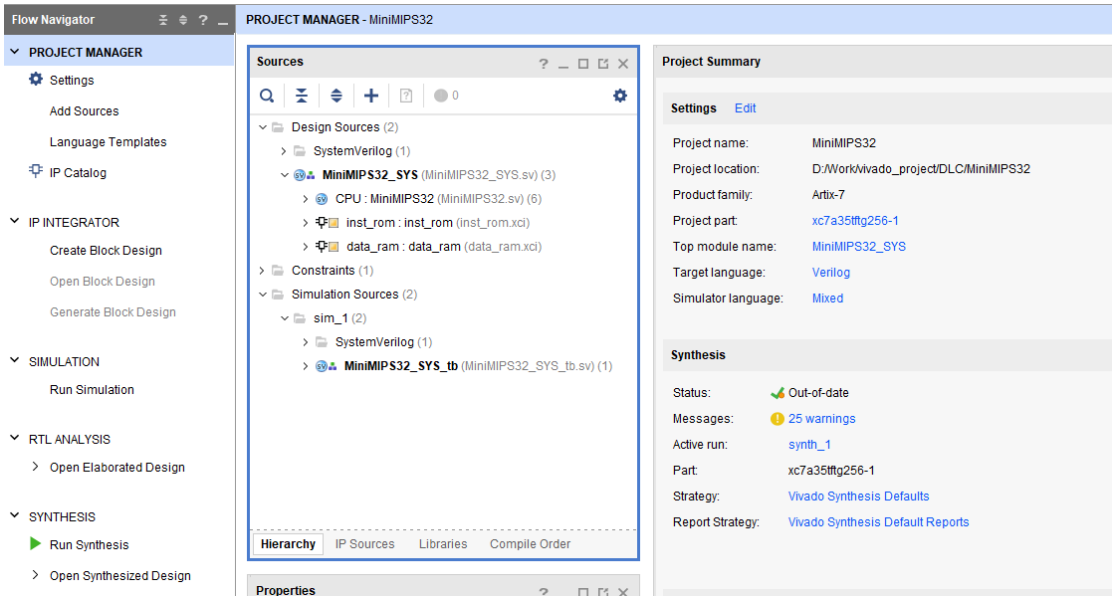


图 5-5 工程 MiniMIPS32

(2). 在文件 MiniMIPS32.sv 中，已经根据顶层文件对处理器的例化形式完成了单周期 MIPS 处理器模块的定义。同学们在该文件中实现处理器的所有功能。建议分 4 步实现 10 条指令，以便与后续功能仿真一致，便于调试：

第一步：lw 和 sw（访存类指令）

第二步：lui、ori 和 addiu（I-型指令）

第三步：addu 和 slt（R-型指令）

第四步：beq、bne 和 j（转移指令）

步骤 2：功能仿真

(1). 测试用例已经为同学们准备好，位于“**MiniMIPS32\benchmark**”路径下，如图 5-6 所示。每个文件夹对应一个测试用例，请大家按照文件夹编号顺序进行测试，保证测试顺序与步骤 1 中的实现顺序一致。

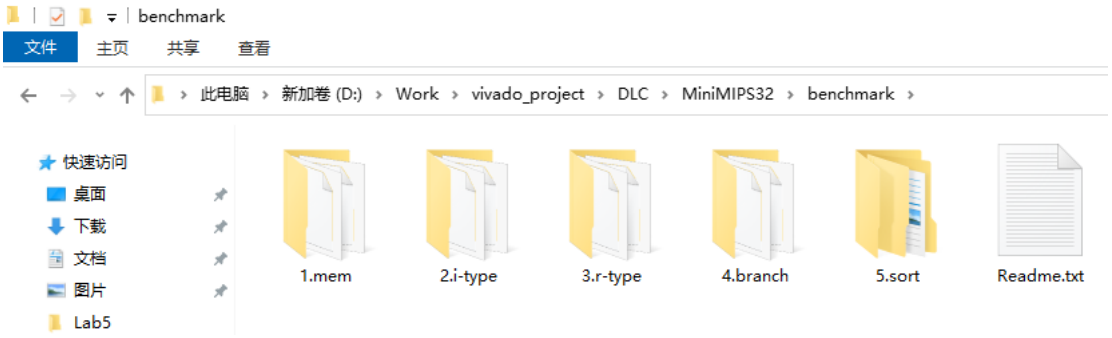
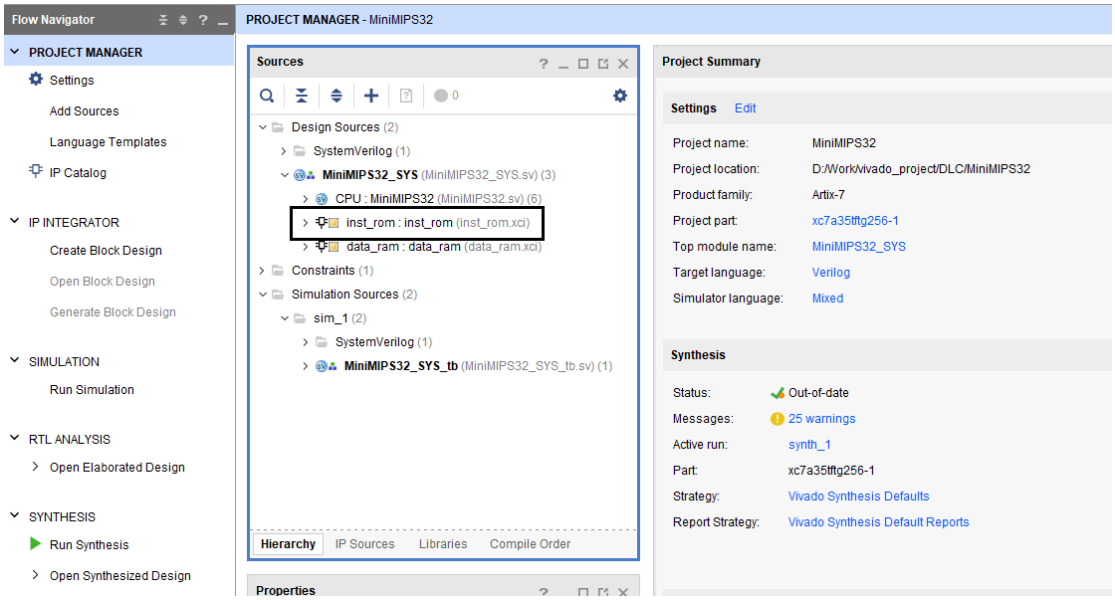


图 测试用例

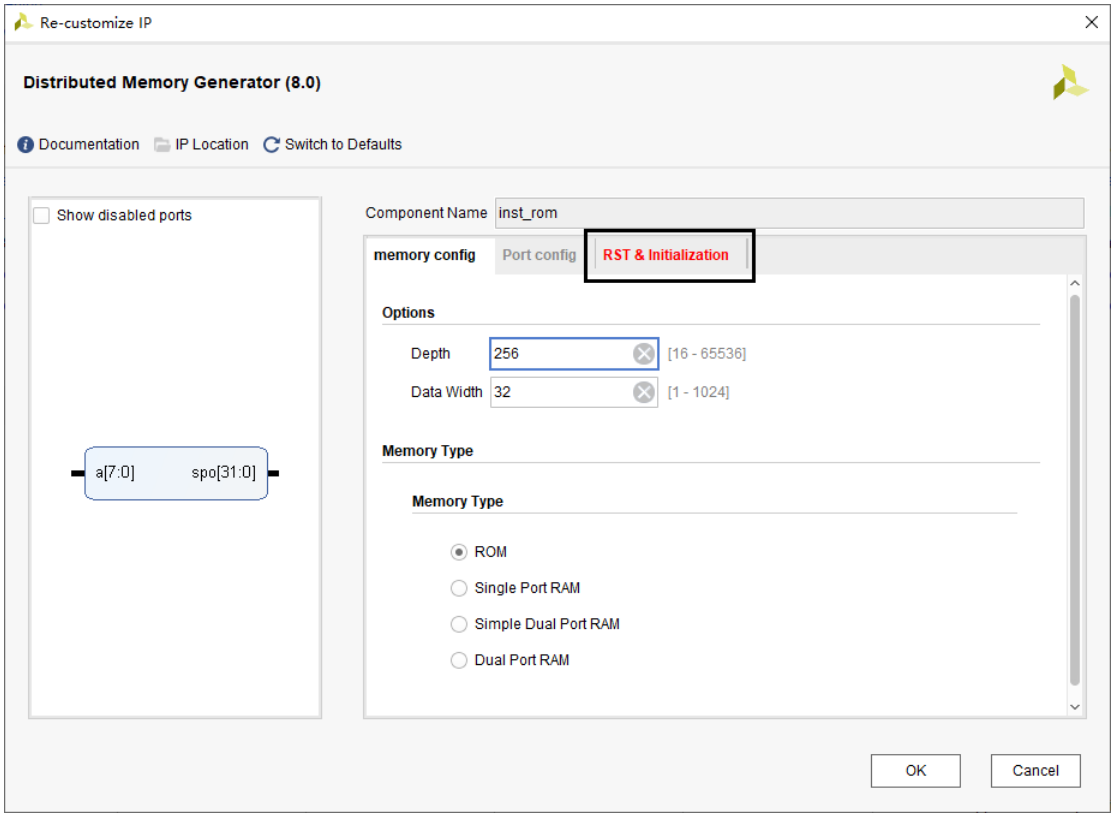
(2). 每个测试用例所在文件夹提供如下文件，其中两个 coe 文件将被分别加载到指令存储器 inst_rom 和 data_ram 中。所有文件均为文本文件，可使用任意文本编辑器打开查看。

- **XXX.S：**测试用例的汇编代码，可参照该文件了解测试用例的功能。
- **XXX_inst.coe：**测试用例中代码段的机器码（即.text 段）。
- **XXX_data.coe：**测试用例中数据段的机器码（即.data 段），如果测试用例中没有定义.data 段，则不存在该文件。

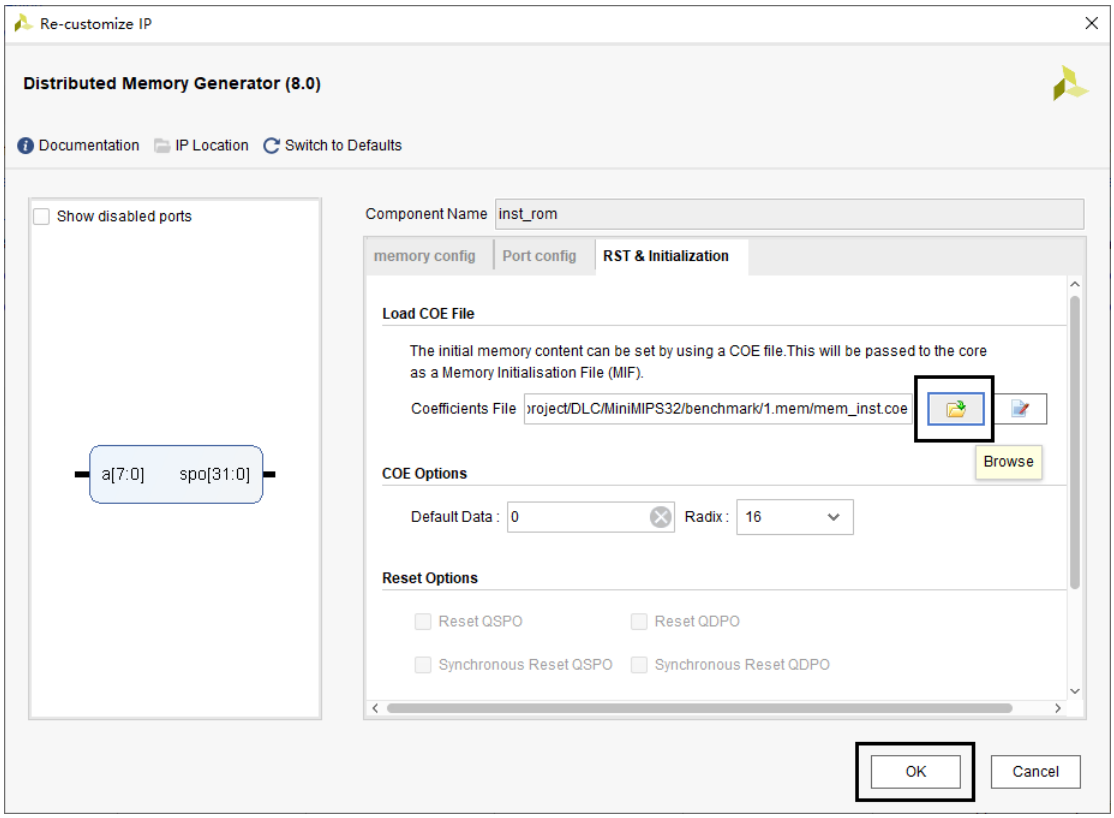
(3). 仿真之前必须加载指令，即加载文件 **XXX_inst.coe**。首先，双击指令存储器 inst_rom，如图 5.7(a)所示。然后，在弹出的窗口中选择标签 RST&Initialization，如图 5.7 (b)所示。接着，点击 Browse 按钮，选择需要加载的 XXX_inst.coe 文件，再点击 OK 按钮完成加载，如图 5.7 (c)所示。在弹出的窗口中，单击 Generate 按钮，更新指令存储器 IP，如图 5.7 (d)所示。


Flow Navigator
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
PROJECT MANAGER - MiniMIPS32
Sources
Design Sources (2)
SystemVerilog (1)
MiniMIPS32_SYS (MiniMIPS32_SYS.sv) (3)
CPU : MiniMIPS32 (MiniMIPS32.sv) (6)
inst_rom : inst_rom (inst_rom.xci)
data_ram : data_ram (data_ram.xci)
Constraints (1)
Simulation Sources (2)
sim_1 (2)
SystemVerilog (1)
MiniMIPS32_SYS_tb (MiniMIPS32_SYS_tb.sv) (1)
Hierarchy
IP Sources
Libraries
Compile Order
Properties
Project Summary
Settings
Edit
Project name:
MiniMIPS32
Project location:
D:/Work/vivado_project/DLC/MiniMIPS32
Product family:
Artix-7
Project part:
xc7a35tftg256-1
Top module name:
MiniMIPS32_SYS
Target language:
Verilog
Simulator language:
Mixed
Synthesis
Status:
Out-of-date
Messages:
25 warnings
Active run:
synth_1
Part:
xc7a35tftg256-1
Strategy:
Vivado Synthesis Defaults
Report Strategy:
Vivado Synthesis Default Reports

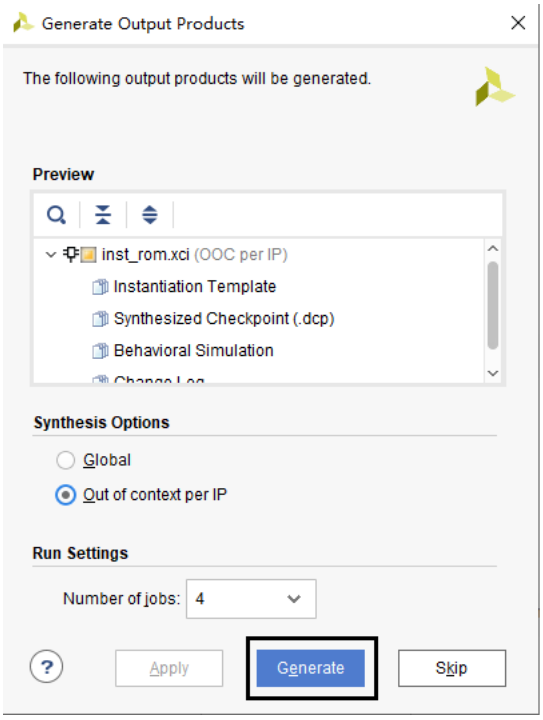
(a)


Re-customize IP
Distributed Memory Generator (8.0)
Documentation
IP Location
Switch to Defaults
Show disabled ports
a[7:0]
spo[31:0]
Component Name
inst_rom
memory config
Port config
RST & Initialization
Options
Depth
256
[16 - 65536]
Data Width
32
[1 - 1024]
Memory Type
Memory Type
ROM
Single Port RAM
Simple Dual Port RAM
Dual Port RAM
OK
Cancel

(b)


Re-customize IP
Distributed Memory Generator (8.0)
Documentation
IP Location
Switch to Defaults
Show disabled ports
a[7:0]
spo[31:0]
Component Name
inst_rom
memory config
Port config
RST & Initialization
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
project/DLC/MiniMIPS32/benchmark/1.mem/mem_inst.coe
Browse
COE Options
Default Data :
0
Radix :
16
Reset Options
Reset QSPO
Reset QDPO
Synchronous Reset QSPO
Synchronous Reset QDPO
OK
Cancel

(c)


Generate Output Products
The following output products will be generated.
Preview
inst_rom.xci (OOC per IP)
Instantiation Template
Synthesized Checkpoint (.dcp)
Behavioral Simulation
Synthesis Options
Global
Out of context per IP
Run Settings
Number of jobs:
4
Apply
Generate
Skip

(d)

图 5-7 加载指令

如果需要加载数据，则双击数据存储器 data_rom，加载 **XXX_data.coe 文件**。后续步骤与加载指令完全相同。上述步骤完成后，即可进行功能仿真。**通过仿真波形图观察寄存器文件中的取值判断程序执行结果是否正确。**

(4). 如果仿真通过，则对工程进行综合、实现、生成 bin 文件，最终下载到远程 FPGA 云平台之上进行板级测试。**注意：远程 FPGA 硬件云平台的验证只能使用“MiniMIPS32\benchmark\5.sort\board”中的两个 coe 文件，其它 coe 文件只能用于功能仿真，不能用于板级验证。**如果验证通过，则 LED 灯 led_g 变为绿色，如图 5-8 所示。

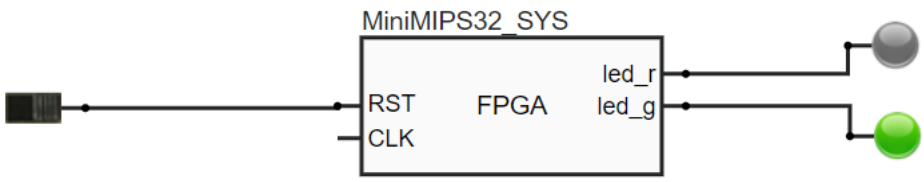


图 5-8 远程 FPGA 硬件云平台的验证

六． 额外说明

如图 5-3 所示，本次实验使用了两个 I/O 外设，即 LED 灯 led_g 和 led_r。相比于 x86 使用设备编号访问 I/O 外设，对于 RISC 处理器而言，I/O 外设和主存通常采用**统一编址**方式，也就是说 I/O 外设地址就是整个系统地址空间的一部分，这样程序员可以便捷地通过调用 **lw 和 sw** 两条指令来处理器和外设之间的通讯。

对于本实验 led_r 的访问地址为 0x80000000，led_g 的访问地址为 0x80040000。假设 led_r 的访问地址已存储在寄存器$a2 中，则通过汇编指令“**sw $a0, 0($a2)**”即可将寄存器$a0 中的值发送给 led_r。

此外，由于 I/O 外设和主存统一编址，因此要特别注意大小端对设计的影响。通常，当 CPU 访问主存的时候需要考虑大小端，但 CPU 访问 I/O 的时候则不区

分大小端，就按通常习惯高有效字节在最左边处理。

七． 实验方式

每位同学独立上机编程实验，实验指导教师现场指导。

八． 参考内容

教材内容和课件

九． 实验报告

1. 画出所实现的单周期 MIPS 处理器原理图。
2. 写出单周期 MIPS 处理器所需的所有控制信号，并通过表格列出每条指令所对应的控制信号的取值。
3. 叙述每条指令在单周期 MIPS 处理器中的执行过程。
4. 给出仿真波形图(仅需要提供一条指令的波形图)，不需要给出板级截图。

九. 附加题

如果按照大端方式存储程序和数据，为了使处理器能够正常运行，需要对其进行哪些修改？请详细说明。