实验三　计算机系统设计与实现

一、实验目标

1. 学习如何构建一个处理器。
2. 学习如何让处理器与外部世界交互。
3. 实现一个完整的计算机系统（含处理器和 I/O）在 FPGA 实验板上运行。

二、软件安装

1. 安装 Xilinx Vivado 软件。安装参考实验一。
2. 安装 Java 运行环境。

从 Oracle 官方下载 x64 compressed archive，解压到一个目录（例如 D:\jdk-21.0.8）。

下载地址：https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip

3. 安装 MIPS 模拟器 MARS。

方式一：从 github 下载并解压。

下载地址：https://github.com/dpetersanderson/MARS/releases/tag/v.4.5.1

方式二：实验附件目录 attach3 或 attach4 下的 Mars.jar 文件。

启动软件的方式：在 windows 的 cmd 终端或 PowerShell 下，运行命令：

D:\jdk-21.0.8\bin\java.exe　-jar　Mars.jar

4. 建议安装微软的 VSCode 编辑代码。

https://code.visualstudio.com/download

5. 建议安装 git 工具管理代码版本

https://git-scm.com/install/windows

三、实验任务

3.1 任务 1：仿真运行

本实验任务是仿真运行一个 MIPS 单周期处理器。相关代码见目录 attch1。

1. 新建 Vivado 项目 task1，芯片型号选择 xc7a35tcpg236-1。
2. 复制 attch1 目录下的文件到 task1 目录下。
3. 将 mipssingle.sv 拆分为两个文件，其中 testbench 脚本保存到 testbench.sv。
4. 在 Design Sources 中添加 mipssingle.sv 文件。

5. 在 Simulation Sources 中添加 testbench.sv 文件和 memfile.dat 文件。

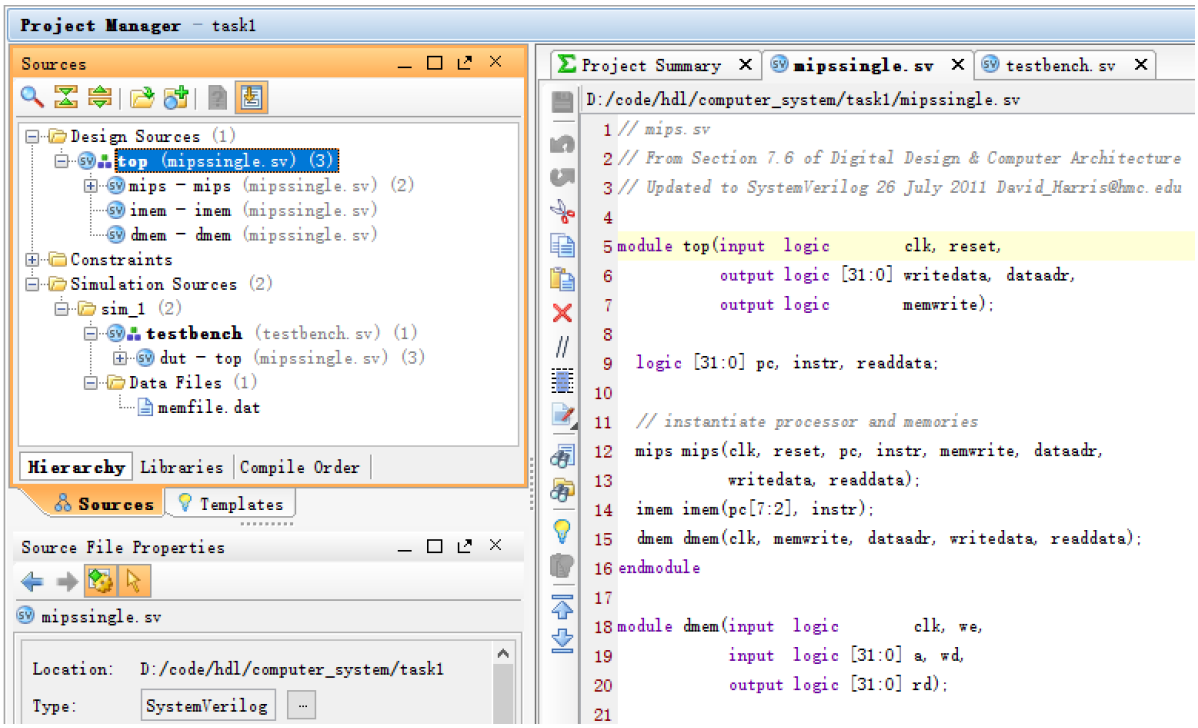
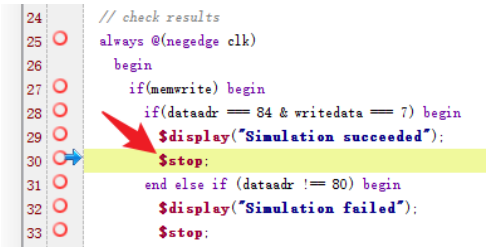
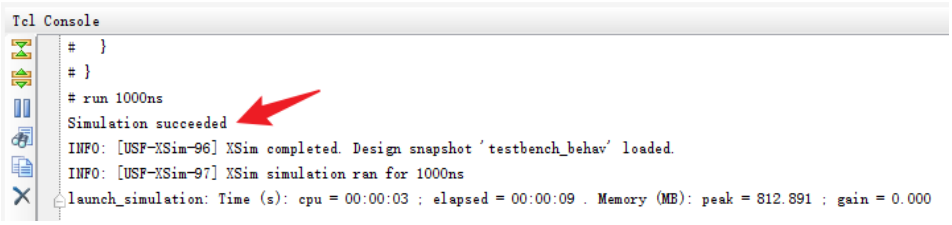


图 1 项目 task1 的 Project manager 内容

6. 运行仿真。正常情况下，Tcl Console 显示 Simulation succeded，testbench.sv 的光标停留在 30 行的$stop 位置。





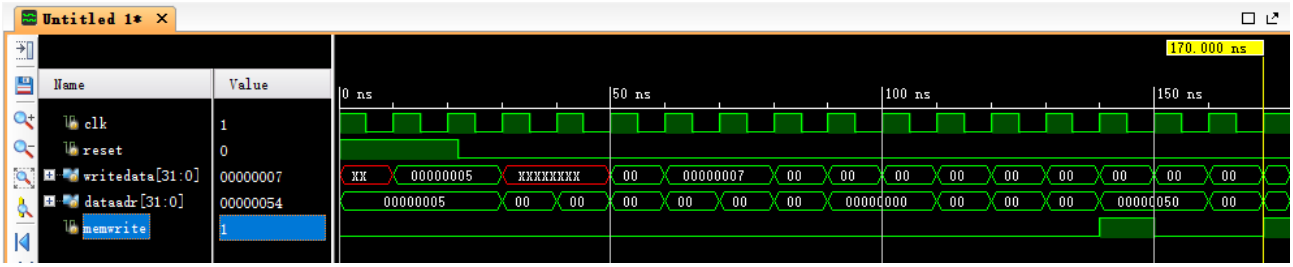


图 2 仿真结果

7. 查看仿真过程中的信号，包括 imem 模块中的地址 a[5:0]和指令 rd[31:0]，alu 模块的 alucontrol[2:0]、a[31:0]、b[31:0]和 result[31:0]。修改数据显示格式 radix，选择合适的观察格式（如 hex 或 unsigned dec 等）。**截图完整的仿真结果，保存到实验报告。**

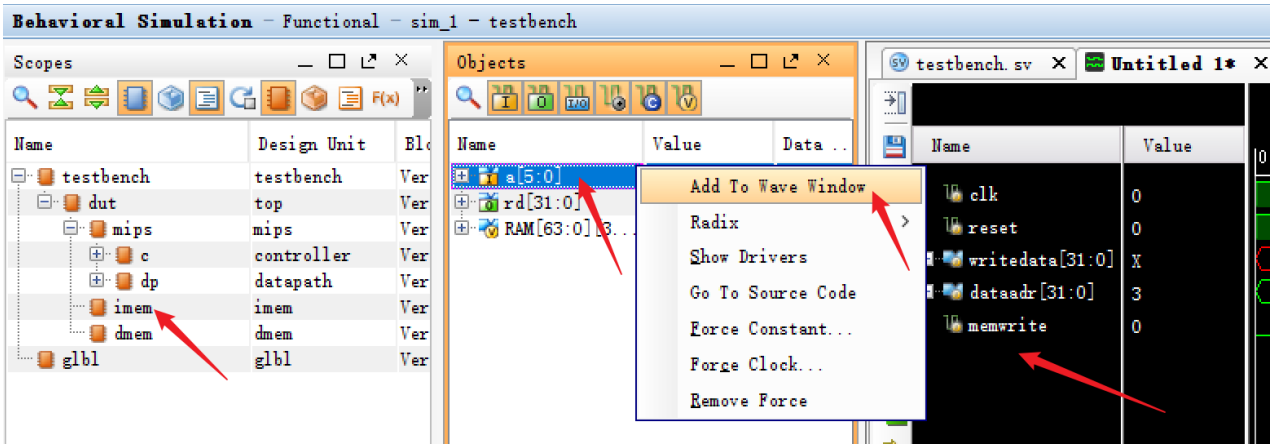


图 3 添加待查看的信号

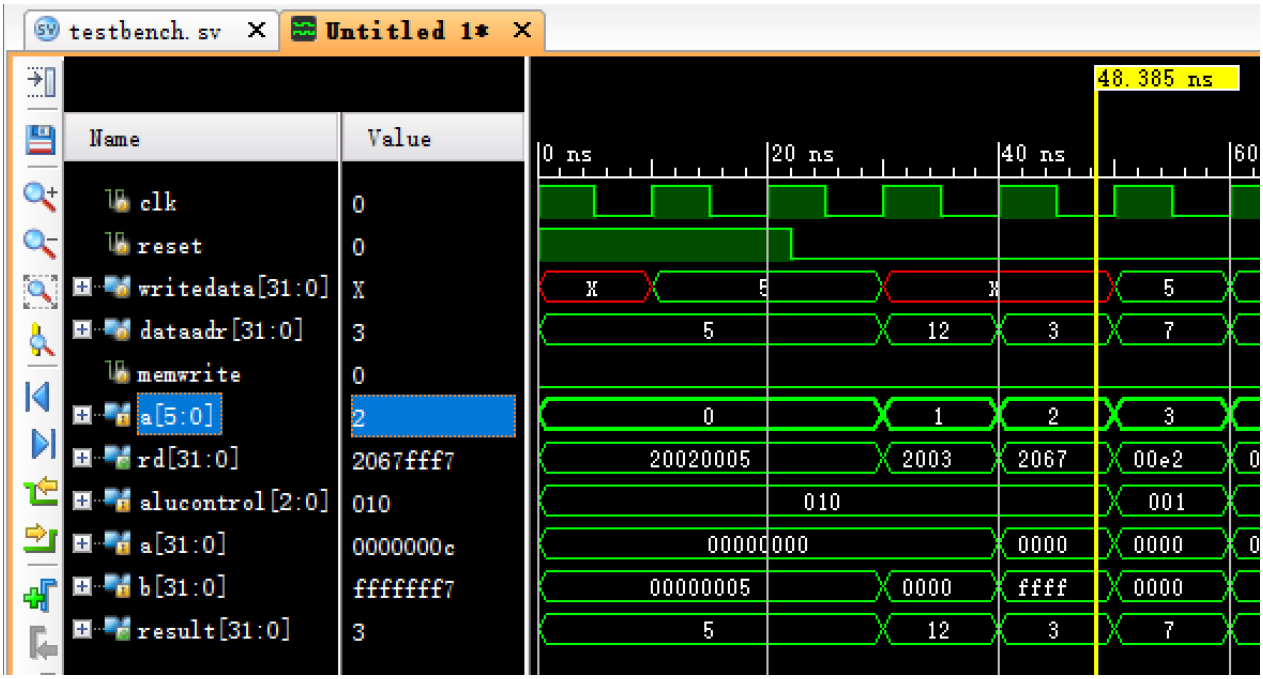


图 4 待观察的信号仿真截图

8. 对照 mipstest.asm 和 memfile.dat，确认仿真中的信号与代码的期望结果一致。例如在 48n 时刻，指令是“addi $7, $3, -9”，机器码是“2067fff7”，a 是 12，b 是-9，操作是加法，alu 结果是 3。**查看 98ns 时刻的信号，写入实验报告。**

mipstest.asm ●

task1 > mipstest.asm

```
7
8   #       Assembly                Description          Address Machine
9   main:   addi $2, $0, 5          # initialize $2 = 5    0       20020005
10          addi $3, $0, 12         # initialize $3 = 12   4       2003000c
11          addi $7, $3, -9         # initialize $7 = 3    8       2067fff7
12          or   $4, $7, $2         # $4 <= 3 or 5 = 7     c       00e22025
13          and  $5, $3, $4         # $5 <= 12 and 7 = 4   10      00642824
14          add  $5, $5, $4         # $5 = 4 + 7 = 11      14      00a42820
15          beq  $5, $7, end        # shouldn't be taken   18      10a7000a
16          slt  $4, $3, $4         # $4 = 12 < 7 = 0      1c      0064202a
17          beq  $4, $0, around     # should be taken      20      10800001
18          addi $5, $0, 0          # shouldn't happen     24      20050000
19  around: slt  $4, $7, $2         # $4 = 3 < 5 = 1       28      00e2202a
20          add  $7, $4, $5         # $7 = 1 + 11 = 12     2c      00853820
21          sub  $7, $7, $2         # $7 = 12 - 5 = 7      30      00e23822
22          sw   $7, 68($3)         # [80] = 7             34      ac670044
23          lw   $2, 80($0)         # $2 = [80] = 7        38      8c020050
24          j    end                # should be taken      3c      08000011
25          addi $2, $0, 1          # shouldn't happen     40      20020001
26  end:    sw   $2, 84($0)         # write adr 84 = 7     44      ac020054
```

图 5　mipstest.asm 汇编代码

9. 关闭 task1 项目，退出 Vivado 软件。在 windows 的 cmd 终端中运行 clean.bat，清除临时文件。最终 task1 目录下的文件如下图，其中 doc 目录用于存放实验报告的截图文件。

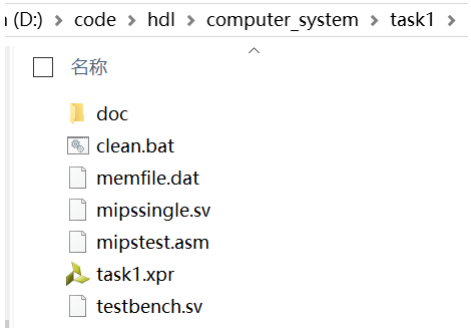


图 6 项目 task1 的文件组成

3.2 任务 2：实物运行

综合任务 1 的代码，部署在实验板的 FPGA 芯片运行。芯片板上的时钟为 100M，通过时钟分频后再输入给 mips 处理器，便于观察结果。

1. 复制项目 1 到项目 2，修改目录为 task2，修改项目名称为 task2.xpr。
2. 复制 attach2 目录下的文件到 task2 目录下。
3. 用 Vivado 软件打开项目 task2。
4. 在 Design Sources 中添加设计文件 mips_on_fpga.sv 和 counter.sv。右键点击 mips_on_fpga 模块 set as top。

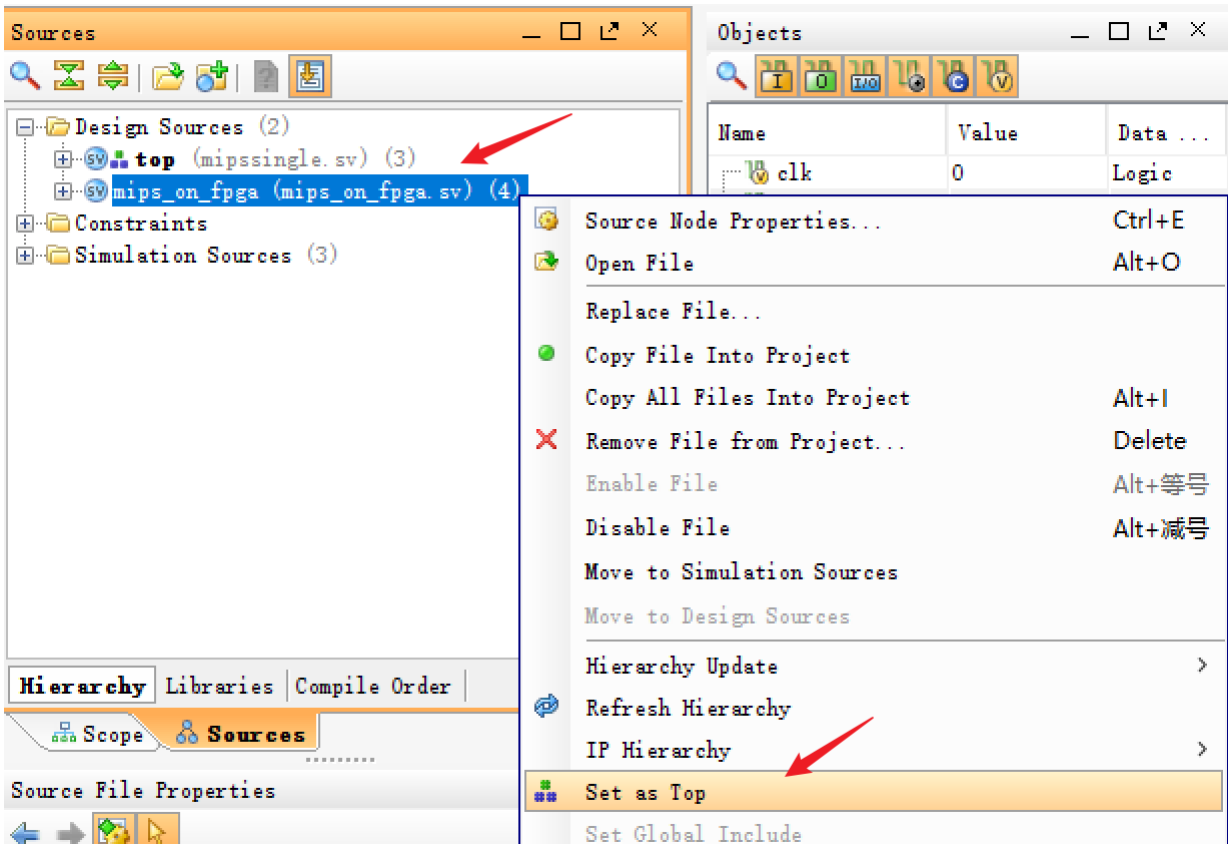


图 7 设置 top 模块

5. 在 Constraints 中添加约束文件 basys3.xdc。实验板设备与 FPGA 管脚对应图 8。

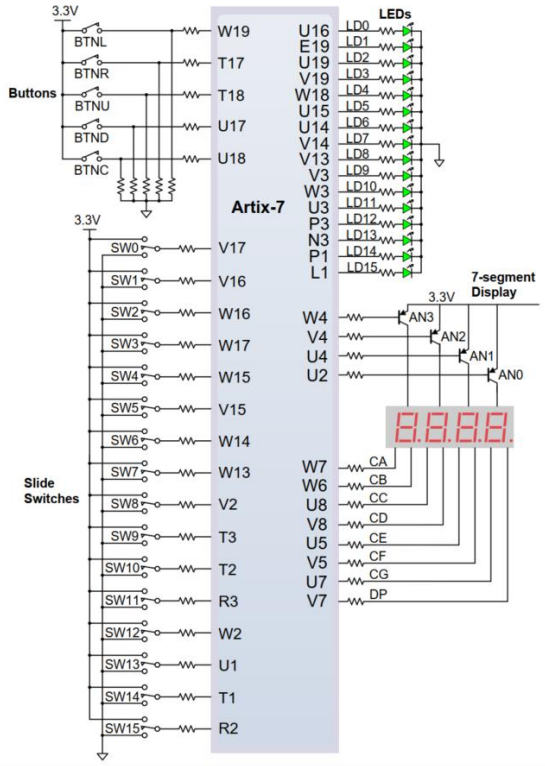


图 8　General purpose I/O devices on the Basys 3

6. 点击 Flow Navigator 中的 Generate Bitstream 生成 bit 文件。系统提示错误。修改 basys3.xdc 文件的 clk 为 fpgaClk，重新生成 bit 文件。

7. 点击 Flow Navigator 中的 Program Device，部署 bit 文件到 FPGA 上运行。

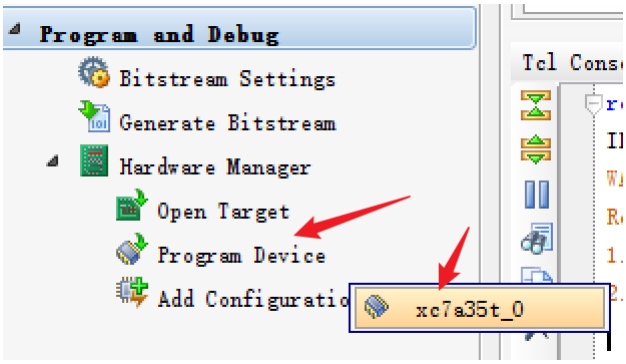


图 9 下载 bit 文件

8. 查看实验板的 LED 灯；持续按住按钮 BTNU，观察 LED 灯。**拍摄图片，描述观察结果，写入实验报告。**

9. 关闭 task2 项目，退出 Vivado 软件。在 windows 的 cmd 终端中运行 clean.bat，清除临时文件。最终 task2 目录下的文件如下图，其中 doc 目录用于存放实验报告的截图文件。

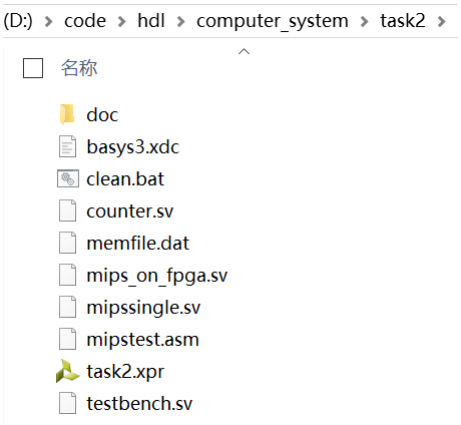


图 10 项目 task2 的文件组成

3.3 任务 3：I/O 交互

本任务实现 MIPS 处理器与外围设备的交互，即处理器读取输入设备（例如开关）的数据，或输出数据到设备（LED 或 7 段码数码管）。系统的架构图如图 11 所示。

第一个输出设备 1 为 LED 组（LD7 到 LD0），设备地址映射地址 0x7FF0；第二个输出设备 2 为 7 段数码管，设备映射地址为 0x7FF1；输入设备为 Switch（SW7 到 SW0），设备映射地址为 0x7FF2。

实验任务主要是编写输入输出设备的译码器信号，根据处理器的输出地址 dataaddr 决定

是读取 IO 设备还是 data memory；综合输出地址 dataaddr 和写信号 memwrite 决定是写 IO 设备还是 data memory。任务的最后步骤是要求**显示学生学号最后四位数字在 7 段数码管上**。

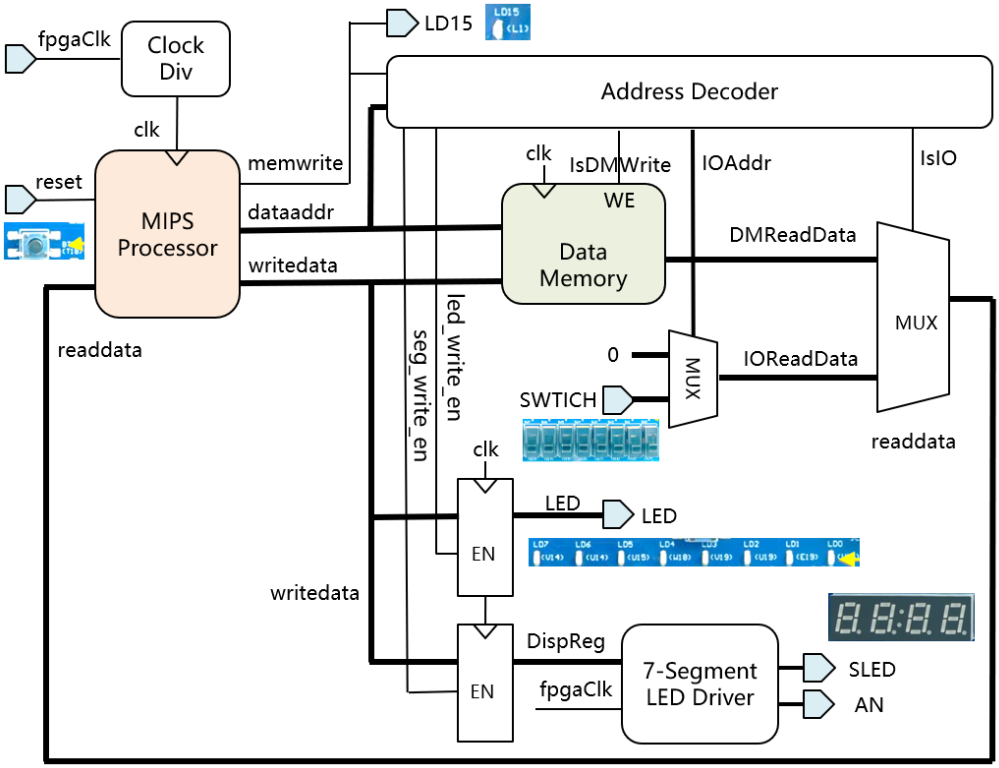


图 11　系统架构图

task3 > ASM task3.asm

```asm
# 数据存储器、IO设备和输入设备测试
.data
pattern: .word 0x6D393E31

.text
main:
        # 测试数码管显示SCUT
        lw   $t0, 0($zero)              # 数码管显示数据
        sw   $t0, 0x7ff1($zero)         # 写入数码管

        # 测试数据存储器
        addi $t0, $zero, 0xFF           # 准备测试数据
        sw   $t0, 0($zero)              # 写入数据存储器
        lw   $t1, 0($zero)              # 读取数据
        sw   $t1, 0x7ff0($zero)         # 输出到LED
        bne  $t0, $t1, fail             # 比较读写结果

# 循环读取开关状态并显示
loop:
        lw   $t5, 0x7ff2($zero)         # 读取开关值
        sw   $t5, 0x7ff0($zero)         # 将开关值输出到LED
        j    loop                       # 无限循环读取开关状态

fail:
        # 测试失败 - 显示0x00
        sw   $zero, 0x7ff0($zero)       # 结果显示在LED上
        j    fail                       # 无限循环
```

图 12　汇编文件 task3.asm

程序（如图 12 所示）首先显示“SCUT”四个字符在输出设备 2（7 段数码管）上，然后不断读取输入设备 1（开关），然后把结果输出到输出设备（LED）。

4 个数码管共用数据线 A、B、C、D、E、F、G 和 DP，芯片轮流输出 0 到 AN3~AN0，用于轮流使能一个数码管。例如，当 AN0 为 0 时，数据线上的数据被输出到 digital 0 上；当 AN1 为 0 时，更改数据线上的数据，输出到 digital1 上。由于 AN 信号切换足够快，依据人眼的视觉暂留效应，人可以看到 4 个显示。详细说明见实验板手册。

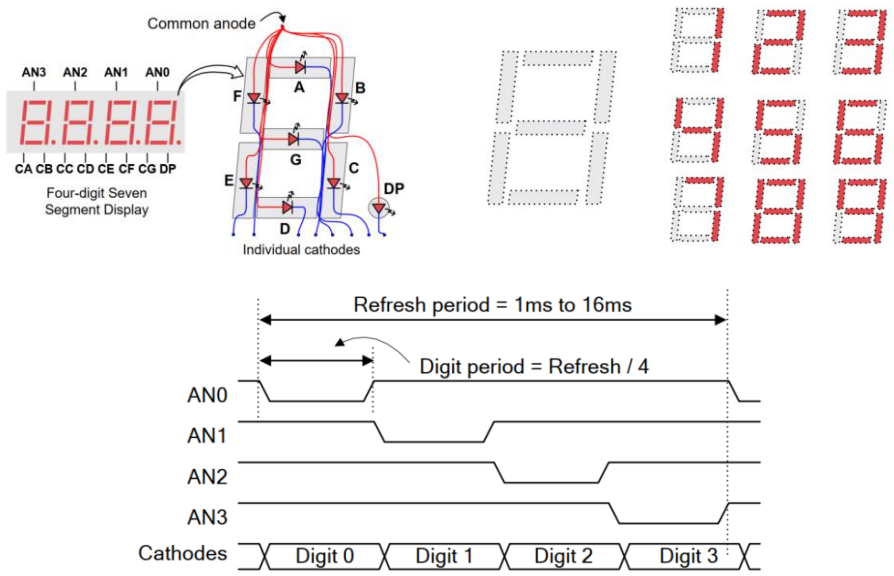


图 13　7 段数码管控制

任务步骤：

1. 复制项目 2 到项目 3，修改目录为 task3，修改项目名称为 task3.xpr。
2. 复制 attach3 目录下的文件到 task3 目录下。
3. 用 Vivado 软件打开项目 task3.xpr。系统提示 syntax error。

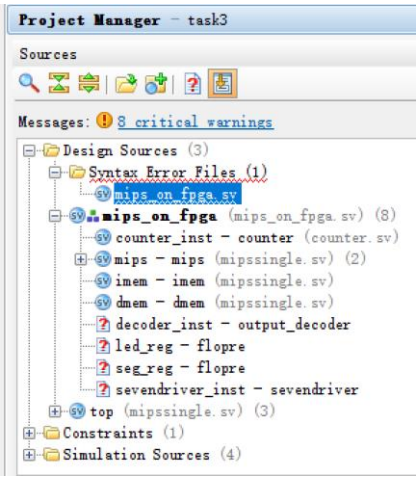
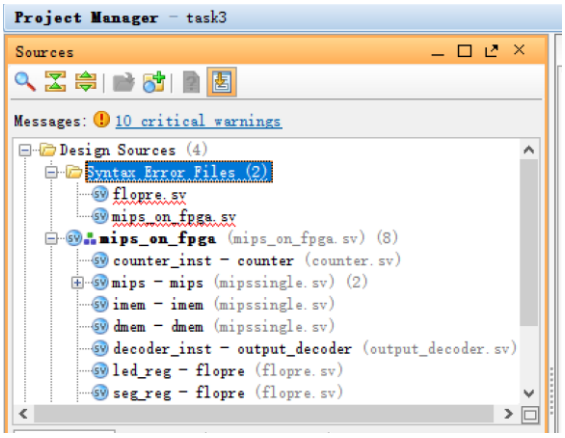


图 14 原始文件的语法错误提示（缺失文件）

4. 在 Design Sources 中添加设计文件 flopre.sv、output_decoder.sv 和 sevendriver.sv。系统依然存在语法错误。



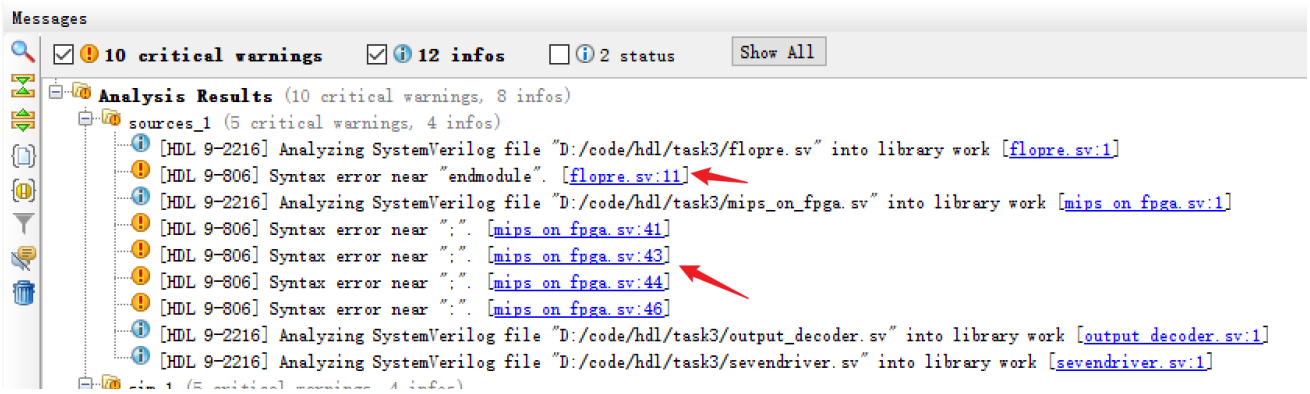


图 15 原始文件的语法错误提示（缺失代码）

5. 查找项目中所有 sv 设计文件中的 TODO 标注（提示：7 个位置），逐一补充修正代码。例如：在 mipssingle.sv 的 25 行插入代码：

```
initial
begin
   $readmemh("datamem.dat", RAM);    // Initialize the array with this content
end
```

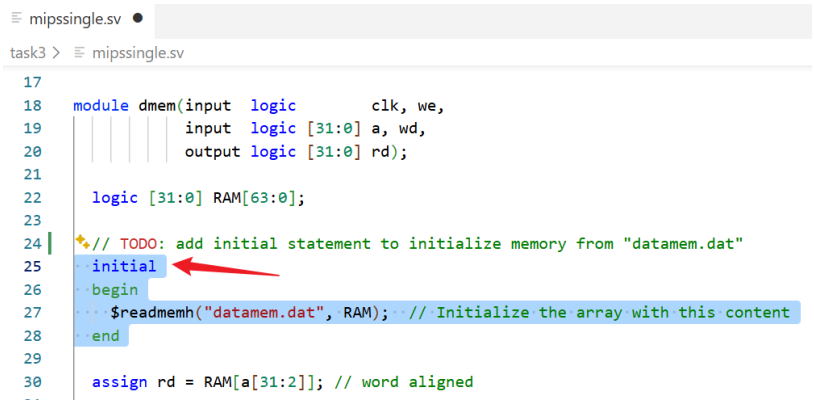


图 16　修正代码中 TODO

6. 查找 xdc 约束文件中的 TODO 标注（提示：1 个位置），补充管脚约束。
7. 在 windows 的 cmd 终端下，运行 mars.bat 启动 Mars 软件。打开汇编 task3.asm。设置 MIPS Memory Configuration 的模式为“Compact, Data at Address 0”，点击“Apply and Close”。

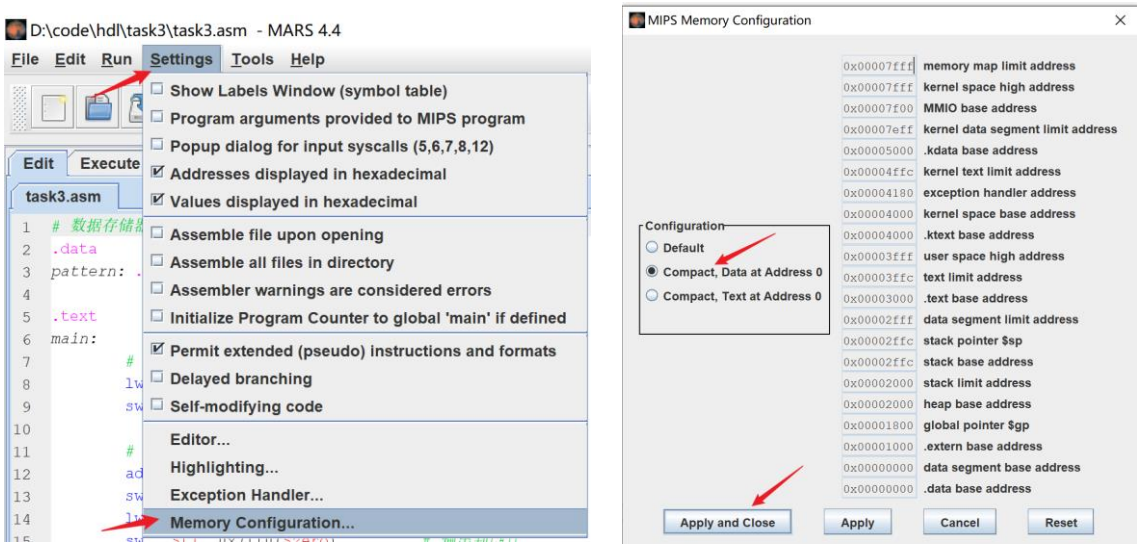


图 17　MIPS Memory Configuration 设置

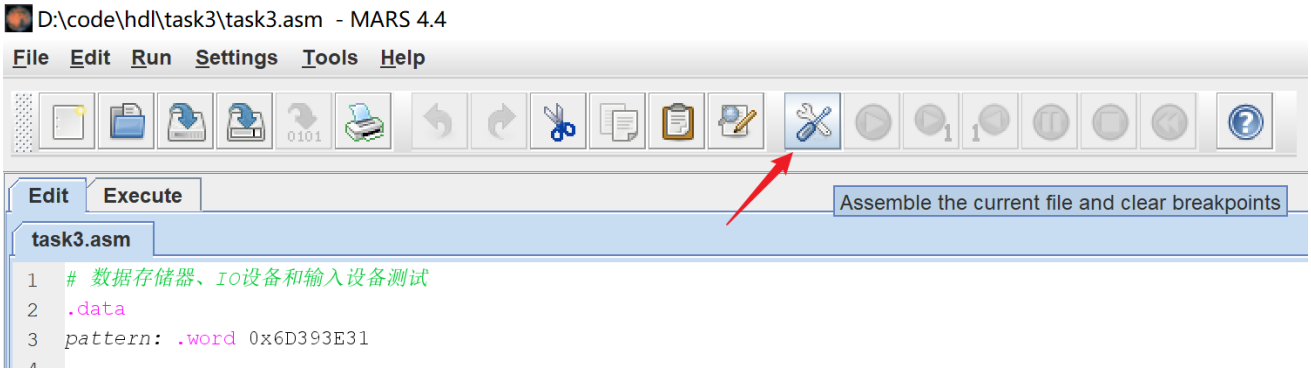
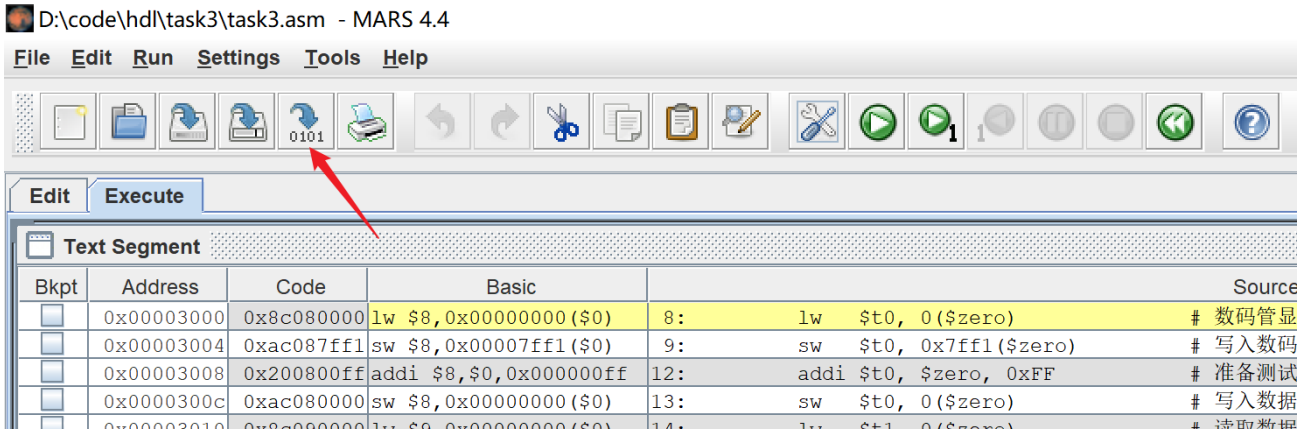


图 18　编译汇编代码



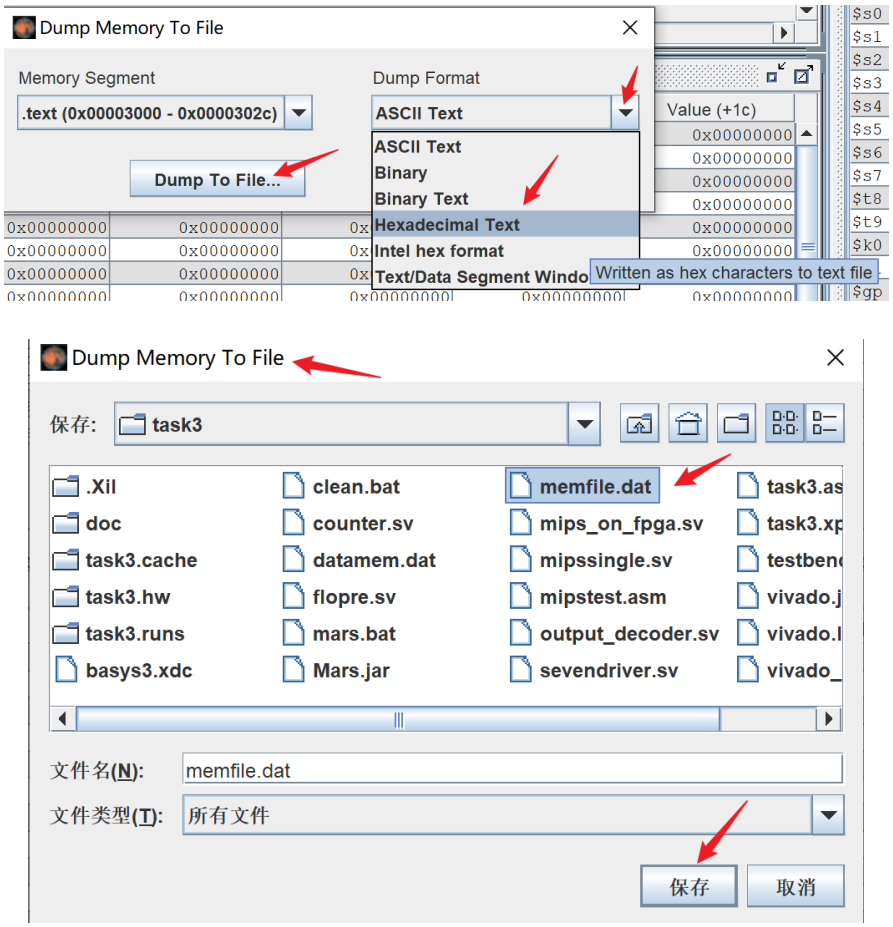


图 19　导出文件 memfile.dat

8. 点击 Flow Navigator 中的 Generate Bitstream 生成 bit 文件。**重要提示：在 Generate Bitstream 之前，先 Reset Synthesis Run（即重新综合项目），确保更新后的 memfile.dat 被项目使用。**

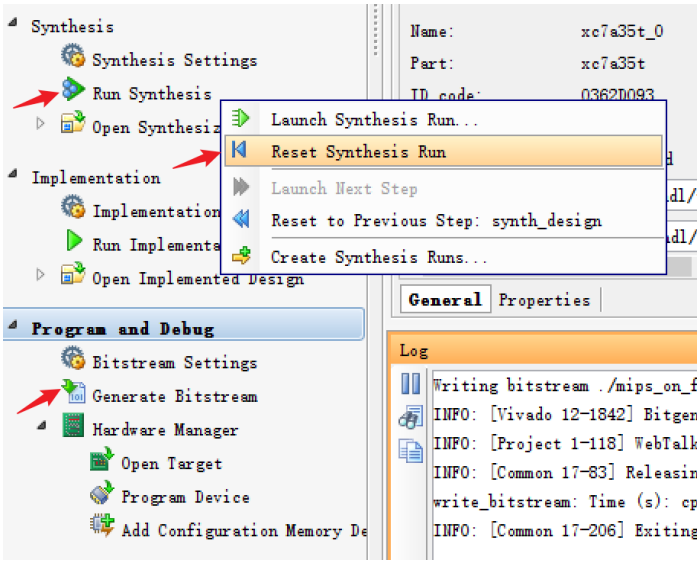


图 20　重新综合后再生成 bit 文件

9. 点击 Flow Navigator 中的 Program Device，部署 bit 文件到 FPGA 上运行。
10. 查看实验板。数码管上显示“SCUT”四个字母；拨动开关 SW0 到 SW7 到 ON 或 OFF，对应 LD0 到 LD7 灯会跟着亮或灭；按下 BTNU（即 reset），重新查看数码管。记录观察结果到实验报告书。

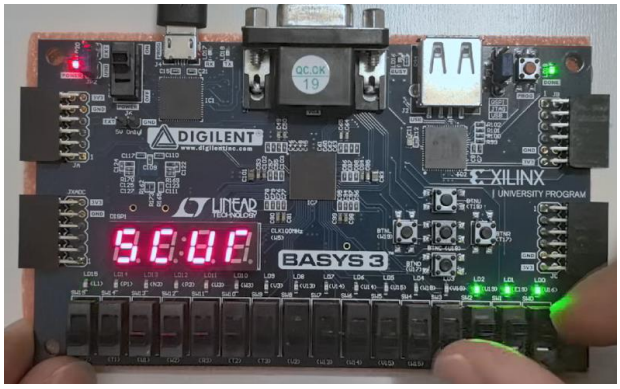


图 21　task3 的实验结果显示

11. 修改数码管显示为学生的学号后四位数字。修改 asm 代码中的 pattern 内容，重新编译 asm 到 memfile.dat。**重要提示：同时要更新 datamem.dat 文件**。重新综合项目，部署到 FPGA，观察和记录实验结果。
12. 关闭 task3 项目，退出 vivado 软件。在 windows 的 cmd 终端中运行 clean.bat，清除临时文件。最终 task3 目录下的文件如下图，其中 doc 目录用于存放实验报告的截图文件。

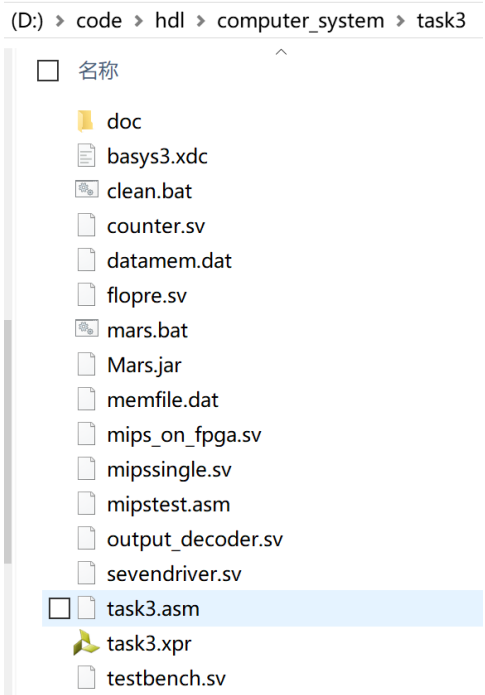


图 22 项目 task3 的文件组成

3.4 任务 4：指令实现

本任务是为 MIPS 处理器增加 SLL（shift left logical）指令的实现。实现方案是修改 ALU、ALU decoder 及相应模块的输入输出端口。

任务步骤：

1. 复制项目 3 到项目 4，修改目录为 task4，修改项目名称为 task4.xpr。
2. 复制 attach4 目录下的文件到 task4 目录下。
3. 用 Vivado 软件打开项目 task4.xpr。
4. 查找项目中 mipssingle.sv 设计文件中的 TODO 标注（提示：2 个位置），逐一补充修正代码。
5. 运行 Mars 软件，打开汇编 task4.asm，编译生成 memfile.dat。
6. 点击 Flow Navigator 中的 Generate Bitstream 生成 bit 文件。**重要提示：在 Generate Bitstream 之前，先 Reset Synthesis Run（即重新综合项目），确保更新后的 memfile.dat 被项目使用。**
7. 点击 Flow Navigator 中的 Program Device，部署 bit 文件到 FPGA 上运行。
8. 查看实验板。“SCUT”四个字母在数码管上向左滚动输出。拍照并记录观察结果到实验报告书。
9. 关闭 task4 项目，退出 vivado 软件。在 windows 的 cmd 终端中运行 clean.bat，清除临时文件。最终 task4 目录下的文件如下图，其中 doc 目录用于存放实验报告的截图文件。

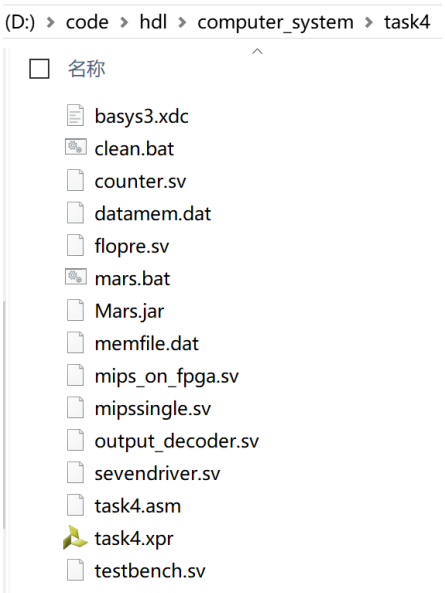


图 23 项目 task4 的文件组成