25A, No.1369,Dongfang Road, Shanghai
Tel:+86-021-58893151 Fax: +86-021-58894598

Lab03: MIPS 处理器部件实现 B

Targeting on Digilent Anvyl

Lab 03: MIPS 处理器部件实现 B

实验简介

本实验旨在使读者实现 MIPS 处理器的部件—Data memory, Instruction memory 和 Registers 三大存储器件。

实验目标

在完成本实验后，您将学会：

- 理解 CPU 的寄存器和内存
- 使用 Verilog 语言设计存储器件
- 使用 ISim 进行行为仿真

实验过程

本实验旨在使读者掌握 MIPS 处理器中内存和寄存器的设计。在本实验中，利用 Verilog HDL 语言描述硬件逻辑实现和仿真内存和寄存器。

实验由以下几个部分组成：

1. Instruction memory 的实现
2. Data Memory 的实现
3. Register 的实现
4. 有符号扩展的实现

Instruction memory 的实现 Step 1

MIPS 的基本架构如图 1 所示，包括 Control，ALU 这样的组合逻辑单元，也包括如 instruction memory，Data memory 和 Registers file 存储单元。本实验主要实现三大存储单元。

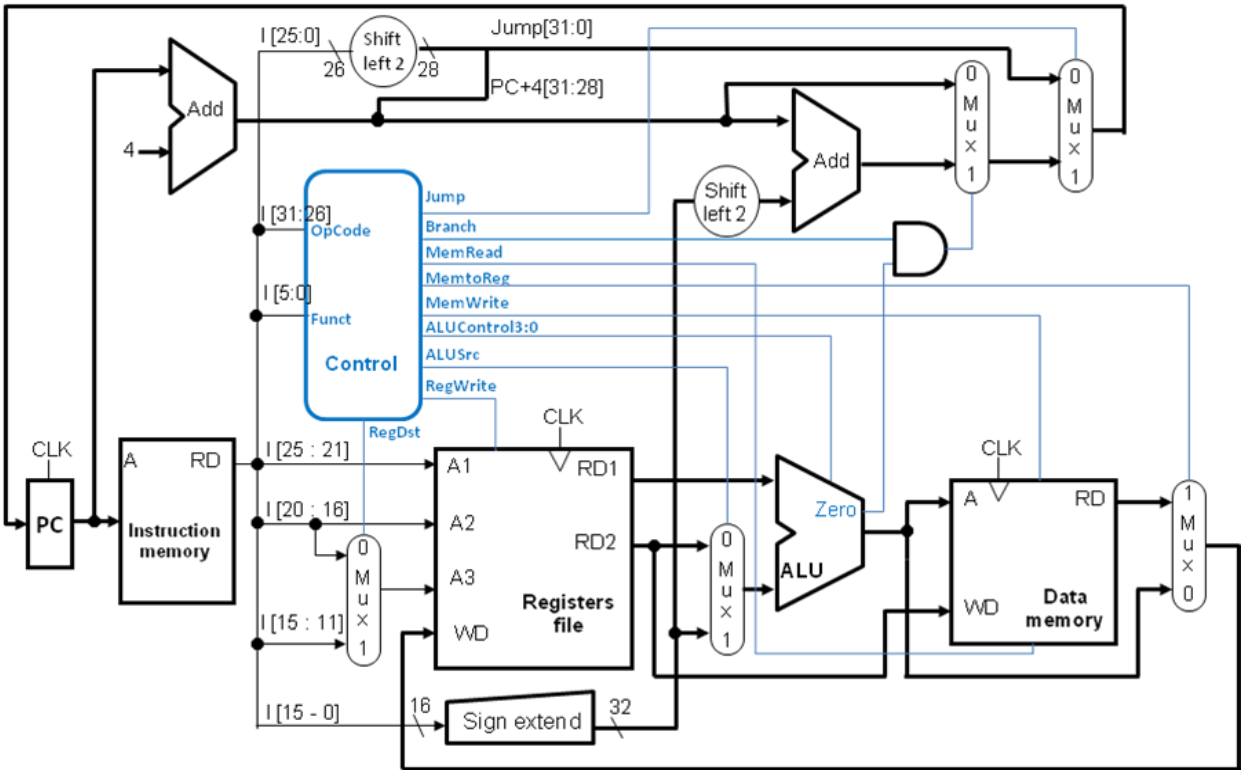

图 1. MIPS 处理器基本架构

处理器指令运行过程可以包括取指令、指令译码、执行、内存操作、寄存器回写，这些操作会对三种存储设备进行读或者写，但是不会同时对同一存储设备进行读写。所以为了实现单周期的 MIPS，做这样一个设计，Instruction Memory 用组合逻辑实现，完成类似于 ROM 的功能，仅作读操作；而 Data memory 和 Register 的读操作用组合逻辑实现，而写操作用时序逻辑来实现。

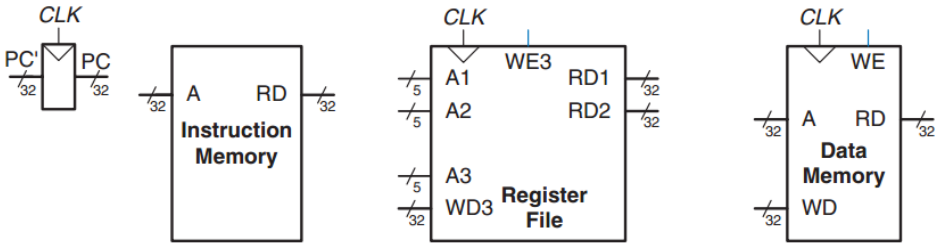

图 2. MIPS 存储设备

- 打开 ISE 工具，新建工程**(注：本实验使用 ISE 13.4)**

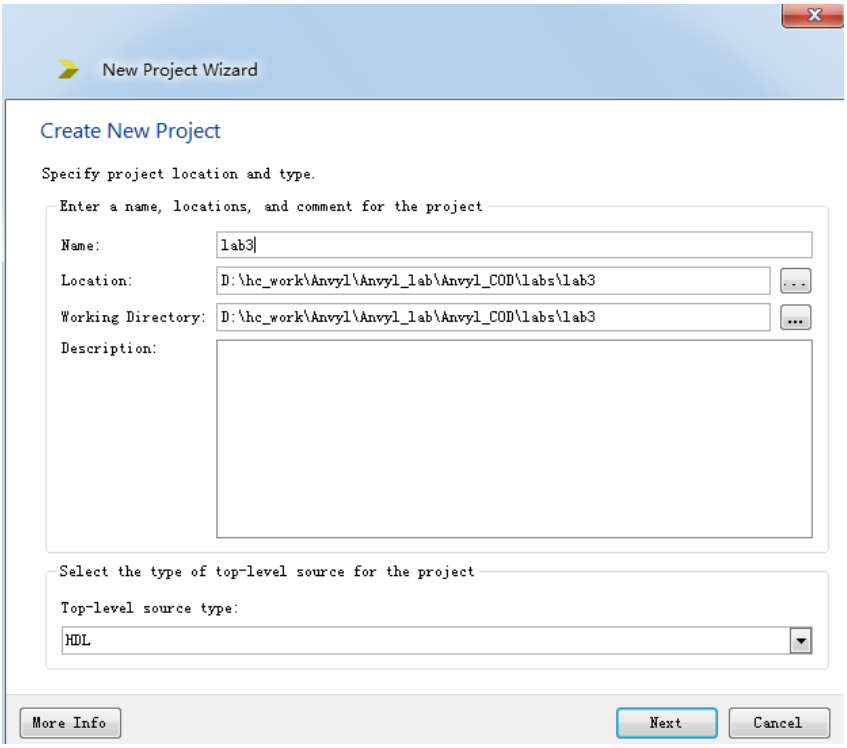

图 3. 创建新工程

- 选择 FPGA 型号、综合和仿真工具、推荐描述语言等配置

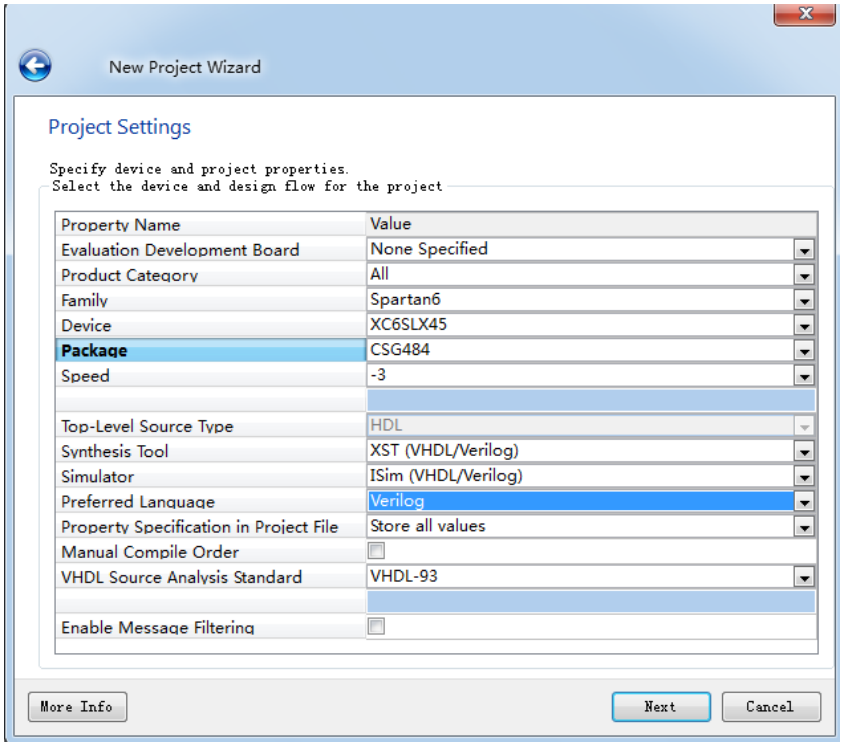

图 4. 新工程设置

- 创建 Data_memory.v 文件，其中 readmemh 用作 ROM 的初始化使用。

```
21  module Instruction_memory(
22      input [31:0] ImemRdAddr,
23      output reg [31:0] Instruction
24      );
25
26  reg [31:0] InstMem [0:255];   //memory space for storing instructions
27
28  //initial the instruction and data memory
29  initial
30  begin
31     $readmemh("Instruction",InstMem,8'h0);
32  end
33
34  always @(ImemRdAddr)
35  begin
36     Instruction <= InstMem[ImemRdAddr];
37  end
38  endmodule
```

图 5. Instruction Memory Verilog 描述

- 添加 Testbench 仿真文件进行行为级仿真。
- 添加激励，即输入信号的控制。

```
33     // Instantiate the Unit Under Test (UUT)
34     Instruction_memory uut (
35        .ImemRdAddr(ImemRdAddr),
36        .Instruction(Instruction)
37     );
38
39  reg [7:0] index;
40    initial begin
41       // Initialize Inputs
42       ImemRdAddr = 0;
43       index =0;
44       // Wait 100 ns for global reset to finish
45       #10;
46       for(index = 0;index<=255;index = index +1)
47          begin
48          #10 ImemRdAddr <= index ;
49          end
50
51       // Add stimulus here
52
```

图 6. 添加输入信号的激励

- 打开 ISim 进行仿真，观察波形，查看仿真结果，是否满足当初的设计。如果有错，检查代码，重新仿真。

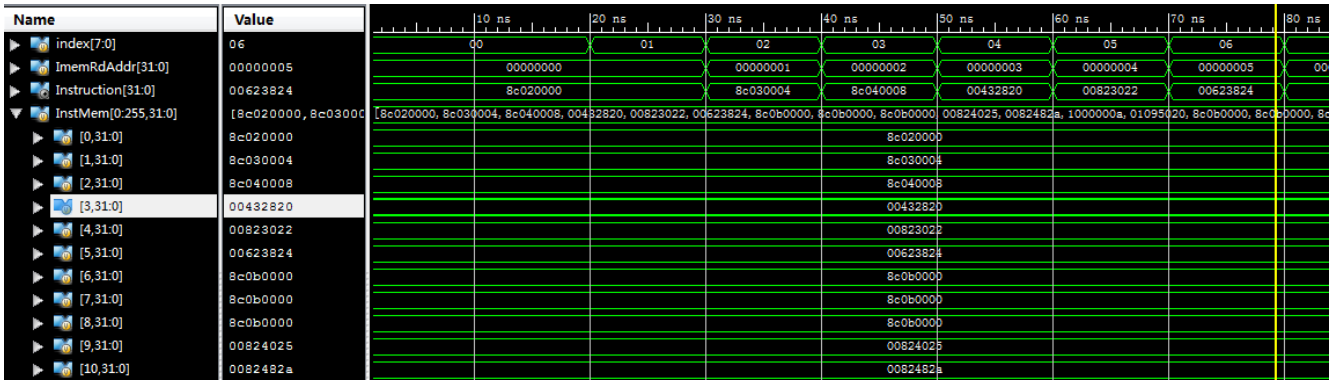

图 7. 仿真波形

Data Memory 的实现 Step 2

Data memory 是用来存储运行完成的数据，或者初始化的数据。其中用于控制 Data memory 的读写信号，可以由一个信号来控制，高低电平控制读写，即图 14 所描述的；或者由图 1 所绘的，分别用两个信号来控制读写。

- 创建 Data_memory.v 文件。

```
20  //////////////////////////////////////////////////////////////////
21  module Data_memory(
22      input Clk,
23      input [31:0] DmemAddr,
24      output [31:0] DmemRdData,
25      input DmemWrite,
26      input [31:0] DmemWrData
27      );
28
29  reg [31:0] DataMem [0:255];   //memory space: 256*32bits
30
31  //initial the instruction and data memory
32  initial
33  begin
34     $readmemh("Data",DataMem,10'h0);
35  end
36
37  always @ (posedge Clk)
38  begin
39      if(DmemWrite == 1'b1)
40         DataMem[DmemAddr] <= DmemWrData;
41  end
42
43
44  assign   DmemRdData = (DmemWrite == 1'b0)? DataMem[DmemAddr]:0;
```

图 8. Data meory Verilog 描述

- 创建 Data_memory_tb.v 测试文件，添加激励信号，进行行为仿真。

```
45  reg [7:0] index;
46     initial begin
47        // Initialize Inputs
48        Clk = 0;
49        DmemAddr = 0;
50        DmemWrData = 0;
51        DmemWrite = 1;
52        //write data into memory
53        for(index = 0;index<=7;index = index +1)
54           begin
55           #10;
56           DmemAddr <= index ;
57           DmemWrData <= index;
58           end
59        //read data from memory
60        DmemWrite = 0;
61        for(index = 0;index<=7;index = index +1)
62           begin
63           #10;
64           DmemAddr <= index ;
65           end
66     end
67  //Clock Generator
68  always #2 Clk = !Clk;
```

图 9. 添加激励信号

- 打开 ISim 进行仿真，观察波形，查看仿真结果，是否满足当初的设计。如果有错，检查代码，重新仿真。

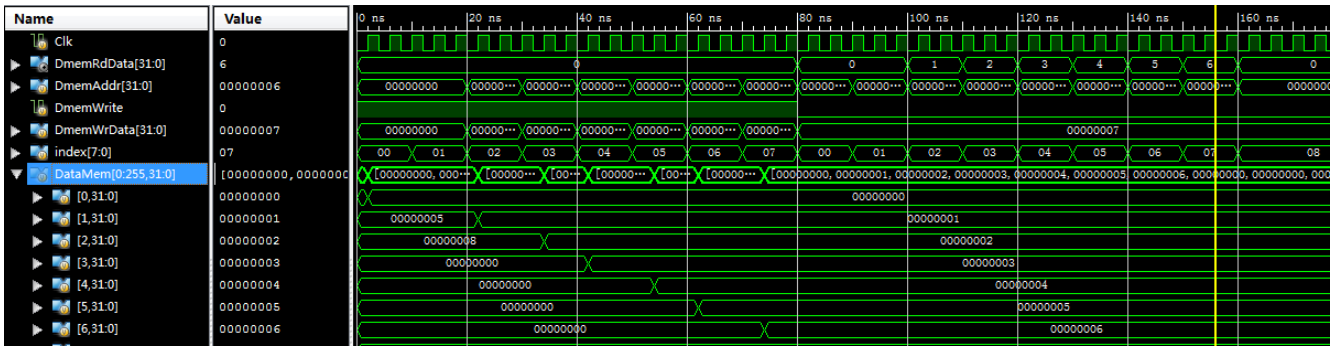

图 10. Data memory 仿真结果

Register 的实现 Step 3

MIPS 处理器有 32 个寄存器，每个寄存器均是 32bit。用作数据的缓存。

- 创建 register.v 文件。由于 register 的设计与 Data memory 区别不大，所以这里不作详述。

```
20  //////////////////////////////////////////////////////////////////////////////////
21  module register(
22      input Clk,
23      input [4:0] RegARdAddr,
24      input [4:0] RegBRdAddr,
25      input [4:0] RegWrAddr,
26      input [31:0] RegWrData,
27      input RegWrite,
28      output [31:0] RegARdData,
29      output [31:0] RegBRdData
30      );
31     |
32     reg [31:0] regFile[0:31];  //32 x 32bit registers
33
34  initial
35  begin
36     $readmemh("register",regFile,32'h0);
37  end
38
39     //write on falling clock edge
40     always @ (negedge Clk)
41        if(RegWrite == 1'b1)
42             regFile[RegWrAddr]<= RegWrData;
43
44     assign RegARdData = (RegARdAddr != 0) ? regFile[RegARdAddr] : 0;
45     assign RegBRdData = (RegBRdAddr != 0) ? regFile[RegBRdAddr] : 0;
```

图 11. register Verilog 描述

- 创建 register_tb.v 测试文件，添加激励信号，进行行为仿真。
- 打开 ISim 进行仿真，观察波形，查看仿真结果，是否满足当初的设计。如果有错，检查代码，重新仿真。