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

Lab02: MIPS 处理器部件实现 A

Targeting on Digilent Anvyl

Lab 02: MIPS 处理器部件实现 A

实验简介

本实验旨在使读者实现 MIPS 处理器的部件—控制器和 ALU。

实验目标

在完成本实验后，您将学会：

- 理解 CPU 控制器
- 理解 ALU 的原理
- 使用 Verilog 语言设计 CPU 控制器和 ALU
- 使用 ISim 进行行为仿真

实验过程

本实验旨在使读者掌握 MIPS 处理器中控制器和 ALU 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 CPU 控制器和 ALU。

实验由以下步骤组成：

1. CPU 控制器的实现
2. ALU 的实现

CPU 控制器的实现 Step 1

打开 ISE 工具进行数字逻辑设计。

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 CPU Control 和 ALU 两个部分。

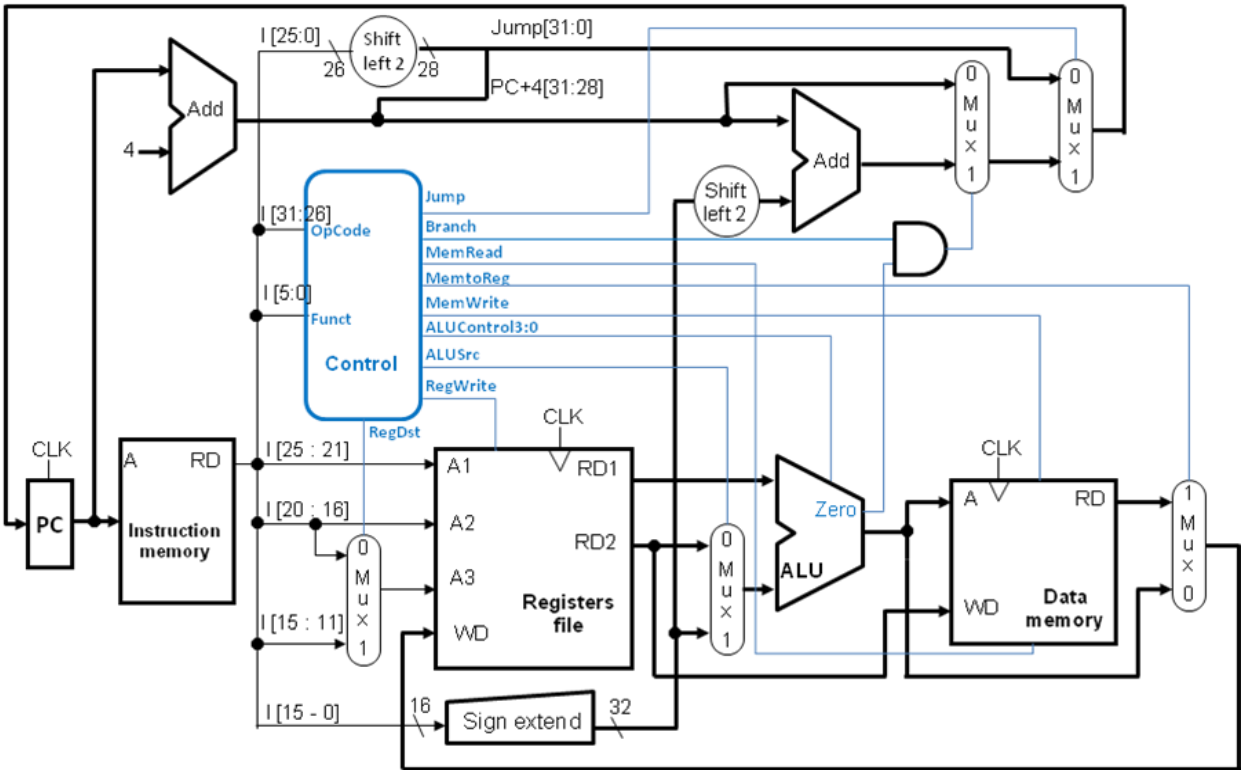


图 1. MIPS 处理器基本架构

CPU Control 单元输入为指令的 opCode 字段，即操作码;以及 R 指令的 funct 编码。操作码和 Funct 编码经过主控制单元的译码，给 ALU，Data Memory，Registers ，Muxs 等部件输出正确的控制信号。

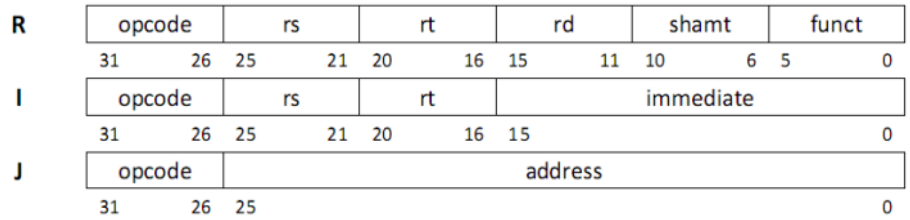


图 2. MIPS 基本指令格式

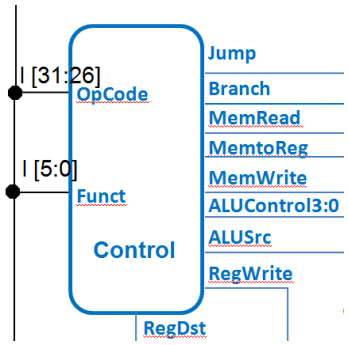


图: 控制模块的 IO 定义

Input or output	Signal name	R-format	lw	sw	beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

注：Jump 指令编码是 000010，Jump 输出信号为 1，其他输出信号都为 0

图 3. OpCode 与控制输出的编码关系

Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	and	0000
R-type	10	OR	100101	or	0001
R-type	10	set on less than	101010	set on less than	0111

图 4. Funct，ALUOp 与 ALU Control 编码关系

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

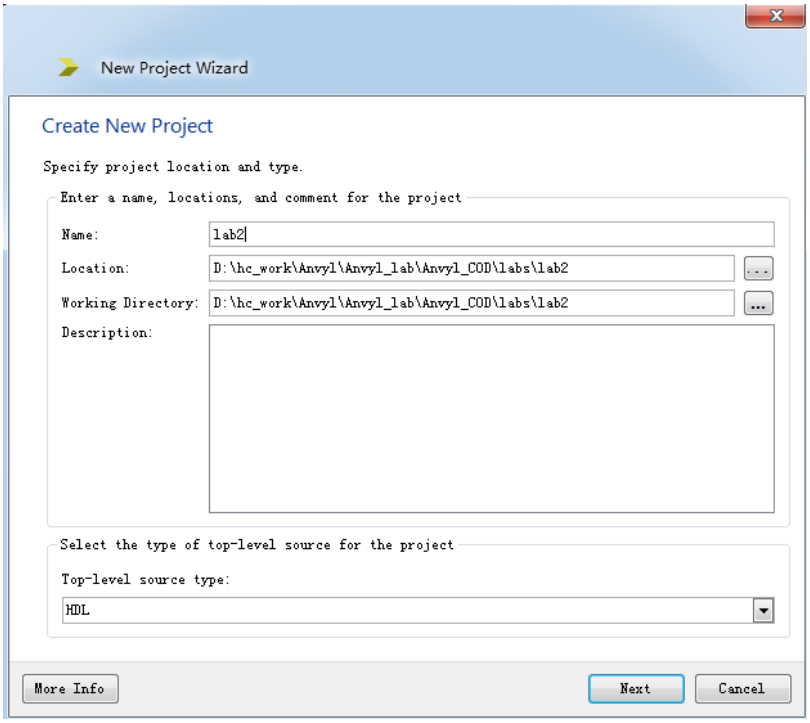


图 5. 创建新工程

- 选择 FPGA 型号、综合和仿真工具、推荐描述语言等配置

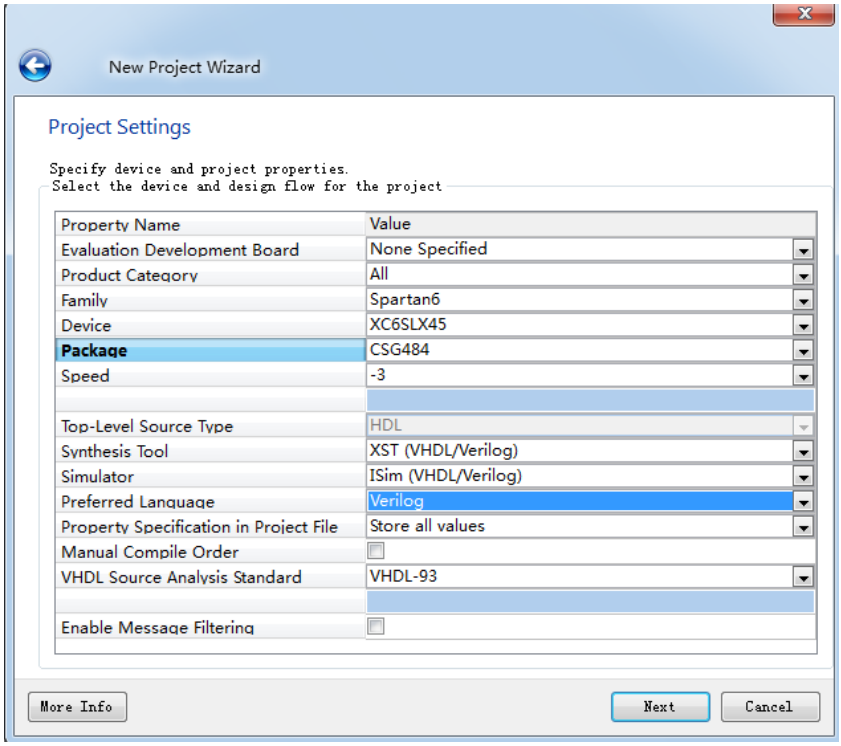


图 6. 新工程设置

- 新建 Verilog 模块文件

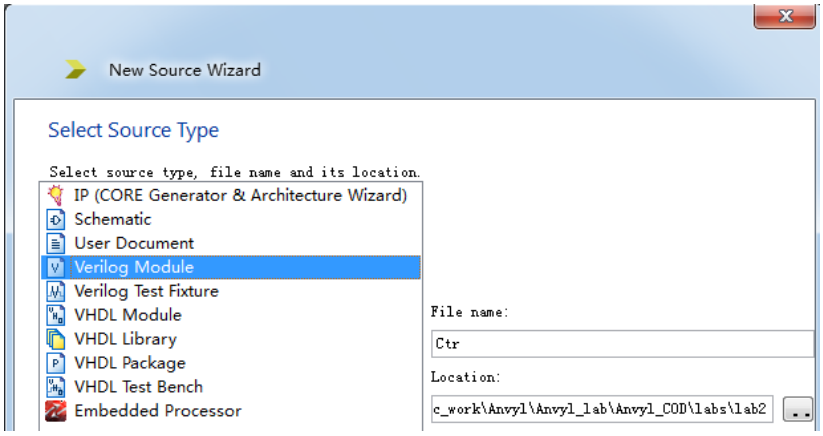
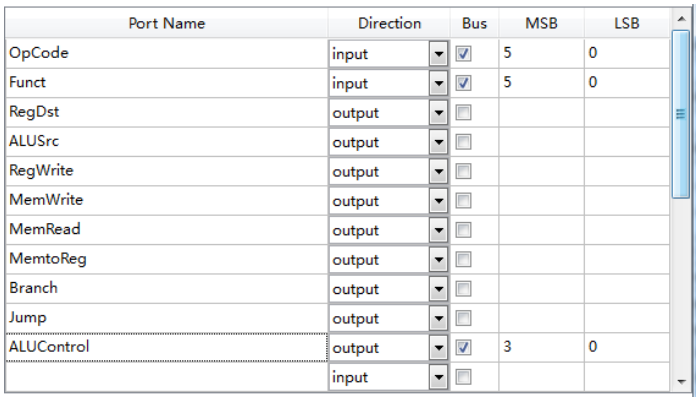


图 7.新建 Verilog 模块

- 定义 Verilog 模块的 I/O 端口,请参考图 2。



Port Name	Direction	Bus	MSB	LSB
OpCode	input	☑	5	0
Funct	input	☑	5	0
RegDst	output	☐		
ALUSrc	output	☐		
RegWrite	output	☐		
MemWrite	output	☐		
MemRead	output	☐		
MemtoReg	output	☐		
Branch	output	☐		
Jump	output	☐		
ALUControl	output	☑	3	0
	input	☐		

图 8.设置模块的 I/O 端口

- 编写 Verilog 代码，输入输出的真值表由图 2 和图 3 给出。在 Verilog 中可以用 case 或者 casex 语句来实现。

```
37  always @(OpCode)
38  begin
39      case(OpCode)
40
41          //R type
42          6'b000000:
43          begin
44              RegDst = 1;
45              ALUSrc = 0;
46              MemToReg = 0;
47              RegWrite = 1;
48              MemWrite = 0;
49              Branch = 0;
50              ALUOp = 2'b10;
51           Jump = 0;
52          end
53
54          //beq
55          6'b000100:
56          begin
57              RegDst = 0;
58              ALUSrc = 0;
```

图 9. case 语句描述

```
121     always @ (ALUOp or Funct )
122     begin
123         casex ({ALUOp,Funct})
124             8'b00xxxxxx : ALUControl = 4'b0010;  //LW :add
125             8'b01xxxxxx : ALUControl = 4'b0110;  //SW :substract
126             8'b1xxx0000 : ALUControl = 4'b0010;  //R-type:add
127             8'b1xxx0010 : ALUControl = 4'b0110;  //R-type:substract
128             8'b1xxx0100 : ALUControl = 4'b0000;  //R-tyep:AND
129             8'b1xxx0101 : ALUControl = 4'b0001;  //R-type:OR
130             8'b1xxx1010 : ALUControl = 4'b0111; //R-type:set on less than
131             default:      ALUControl = 4'b0000;
132         endcase
133     end
```

图 10. casex 语句描述

- 添加 Testbench 仿真文件进行行为级仿真。右键选中 Source 窗口，点击 New Source，创建 Testbench 文件，下一个窗口选择 Ctr 模块，工具会自动为 Ctr 创建 Testbench 模板文件，之后在该文件下添加激励。

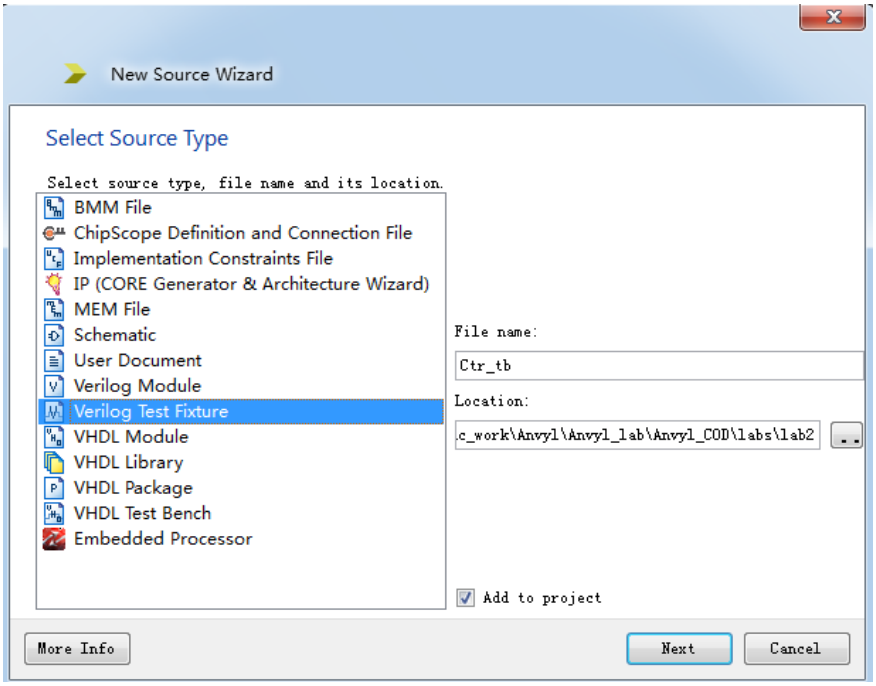


图 11. 创建 Testbench 文件

- 添加激励，即输入信号的控制。

```
57      initial begin
58          // R-type Add
59          OpCode = 6'b000000;
60          Funct = 6'b100000;
61          //R-type Subtract
62          #10;
63          OpCode = 6'b000000;
64          Funct = 6'b100010;
65          //LW
66          #10;
67          OpCode = 6'b100011;
68          Funct = 6'bxxxxxx;
69          //SW
70          #10;
71          OpCode = 6'b101011;
72          Funct = 6'bxxxxxx;
73          //BEQ
```

图 12. 添加输入信号的激励

- 调用 ISim 进行行为仿真，source 窗口上方选择 Simulation，Source 窗口中选择 Ctr_tb 文件，然后在 Process 窗口中选择 Simulate Behavioral Model 开始仿真。

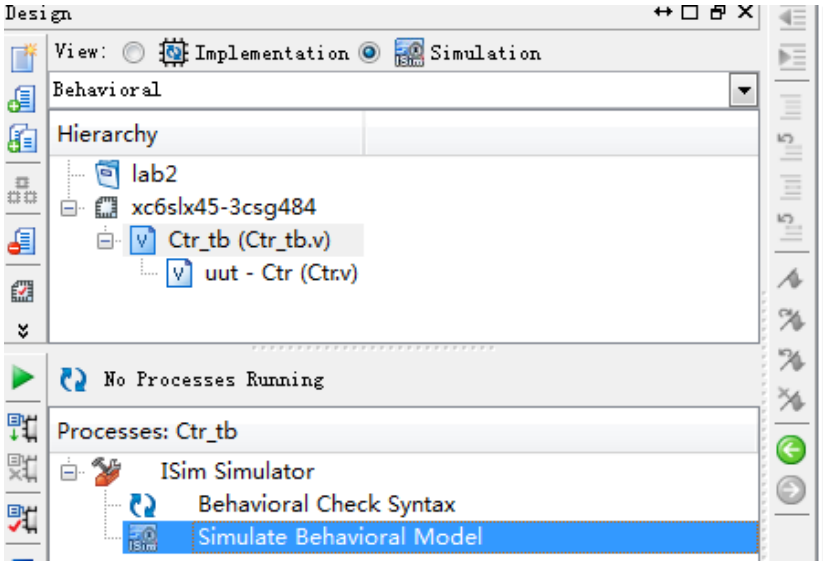


图 13. 调用 ISim 工具

- 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。

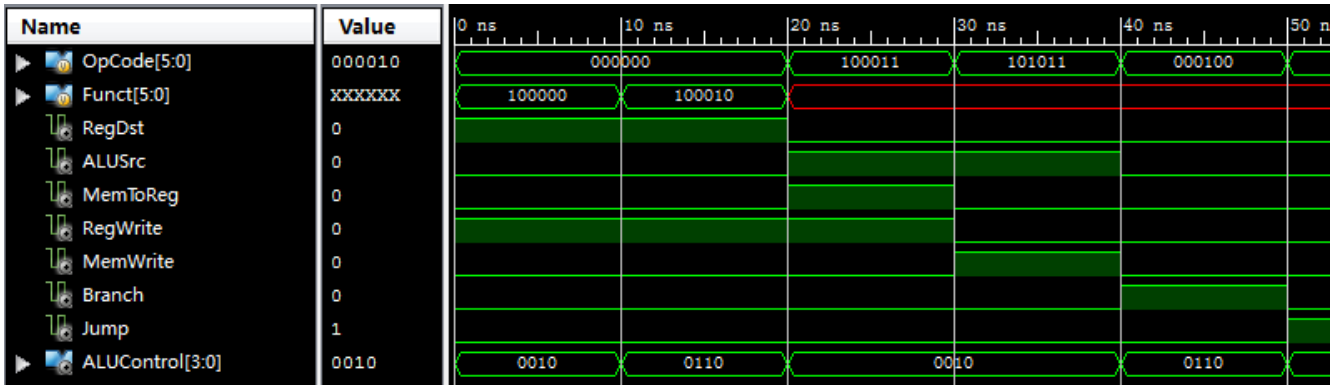


图 14. 仿真波形

ALU 的实现 Step 2

ALU 是 CPU 核心的计算单元，实现诸如加，减，或，与等操作。

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

图 15. 算术操作的编码

- 创建 ALU.v 文件，整个流程与 Ctr 模块的创建和调试一样，这里不再作详述。

```
20 //////////////////////////////////////////////////////////////////////////////////
21 module ALU(
22     input [31:0] SrcA,
23     input [31:0] SrcB,
24     input [3:0] ALUCtr,
25     output Zero,
26     output reg [31:0] ALURes
27     );
28 
29 assign Zero = (ALURes ==1'b0);//Zero is true if ALURes is 0;
30 
31 always @ (SrcA or SrcB or ALUCtr)
32 begin
33     case (ALUCtr)
34        4'b0000: ALURes = SrcA & SrcB;     //AND
35        4'b0001: ALURes = SrcA | SrcB;     //OR
36        4'b0010: ALURes = SrcA + SrcB;     //add
37        4'b0110: ALURes = SrcA - SrcB;     //substract
38        4'b0111: ALURes = SrcA < SrcB ? 1:0;  //set on less than
39        4'b1100: ALURes = ~(SrcA|SrcB);    //NOR
40        default: ALURes = 32'h0;
41     endcase
```

图 16. ALU Verilog 描述

- 创建 ALU_tb.v 测试文件，添加激励信号，进行行为仿真。

```
initial begin
   // AND
   SrcA = 32'hf0f0ffff;
   SrcB = 32'h0000f0f0;
   ALUCtr = 0;
   //OR
   #10;
   ALUCtr = 4'b0001;
   //Add
   #10;
   ALUCtr = 4'b0010;
   //Subtract
   #10;
   ALUCtr = 4'b0110;
   //Set on Less than
   #10;
   ALUCtr = 4'b0111;
   //NOR
   #10;
   ALUCtr = 4'b1100;
   //Others situation
   #10;
   ALUCtr = 4'b1111;
```

图 17. 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。

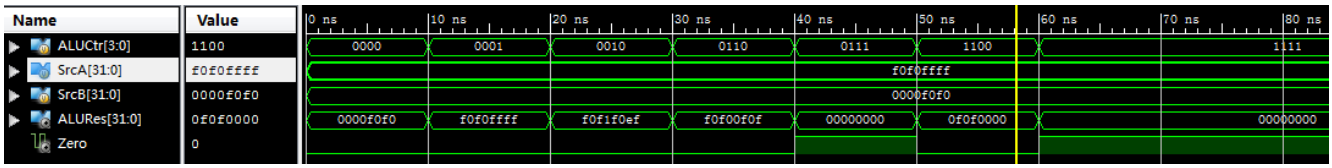


图 18.ALU 仿真结果