Computer Organization & Design实验与课程设计

Lab05-2
流水线处理器—IF、ID设计与集成

Ma De（马德）

made@z ju. edu. cn

2025

College of Computer Science, Zhejiang University

Course Outline

- 一、实验目的
- 二、实验环境
- 三、实验目标及任务

实验目的

1. 理解流水线CPU的基本原理和组织结构

2. 掌握五级流水线的工作过程和设计方法

3. 理解流水线取指、译码的设计原理

4. 设计流水线测试程序

实验环境

- 实验设备

  1. 计算机（Intel Core i5以上，4GB内存以上）系统

  2. NEXYS A7开发板

  3. VIVADO 2017.4及以上开发工具

- 材料

  无

实验目标及任务

- 目标：熟悉RISC-V 五级流水线的工作特点，了解取指、译码的原理，掌握IP核的使用方法，集成并测试CPU

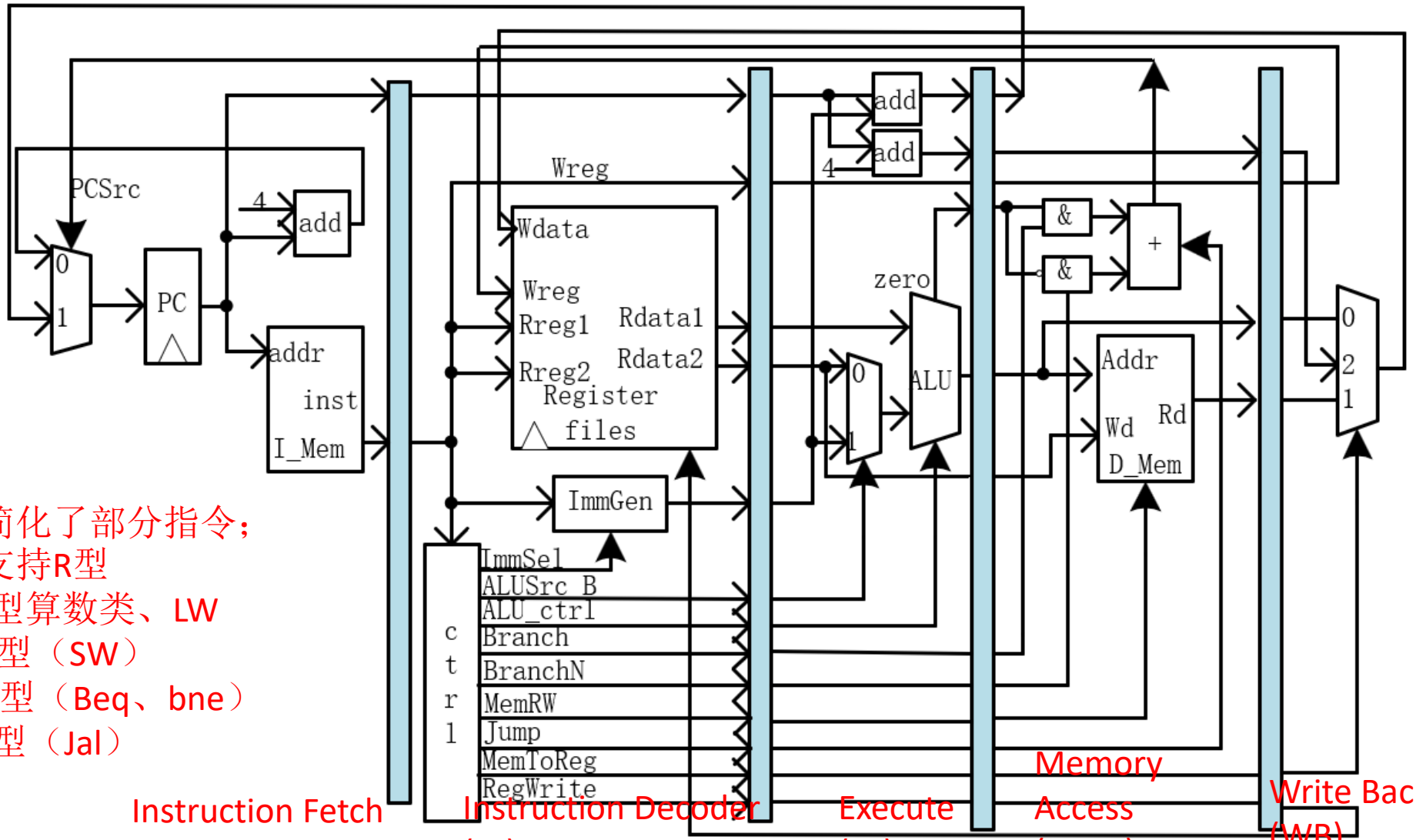
- 任务一：设计取指（IF）、译码（ID）模块，替换实验九的流水线CPU并完成集成
  - 设计取指模块，替换OExp05-1的取指模块并完成集成
  - 设计译码模块，替换OExp05-1的译码模块并完成集成
- 任务二：设计流水线测试方案并完成测试

RISC-V 流水线处理器的原理介绍

------取指（IF）模块介绍

------译码（ID）模块介绍

Pipelined RISC-V RV32I Datapath



简化了部分指令；
支持R型
I型算数类、LW
S型（SW）
B型（Beq、bne）
J型（Jal）

Instruction Fetch (IF)

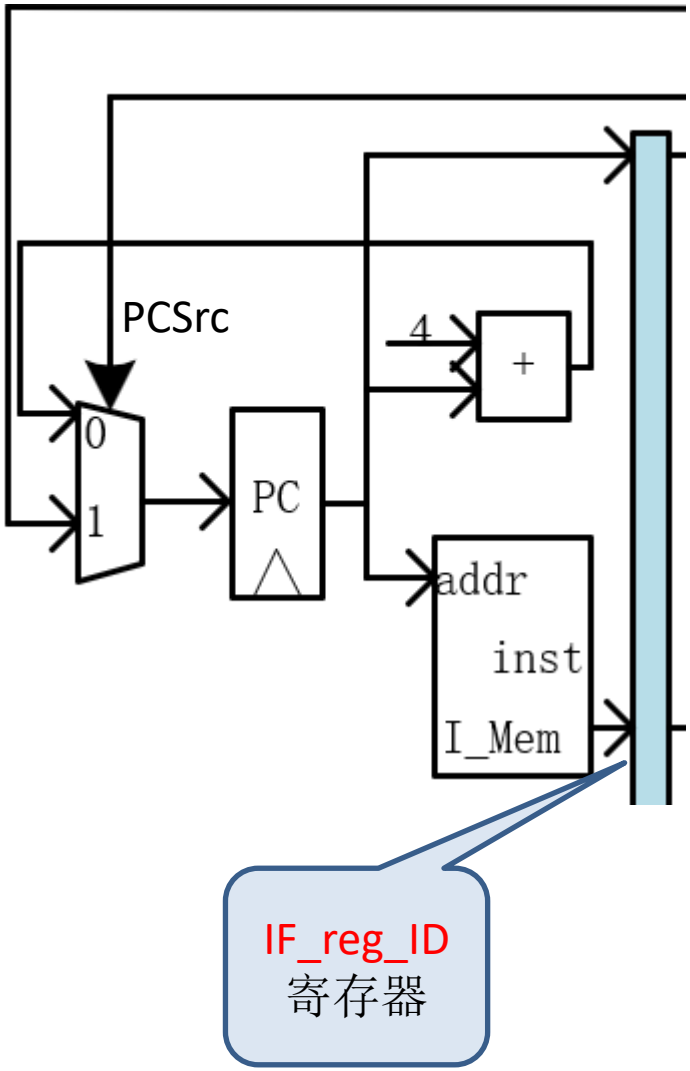
Instruction Decoder (ID)

Execute (Ex)

Memory Access (Mem)

Write Back (WB)

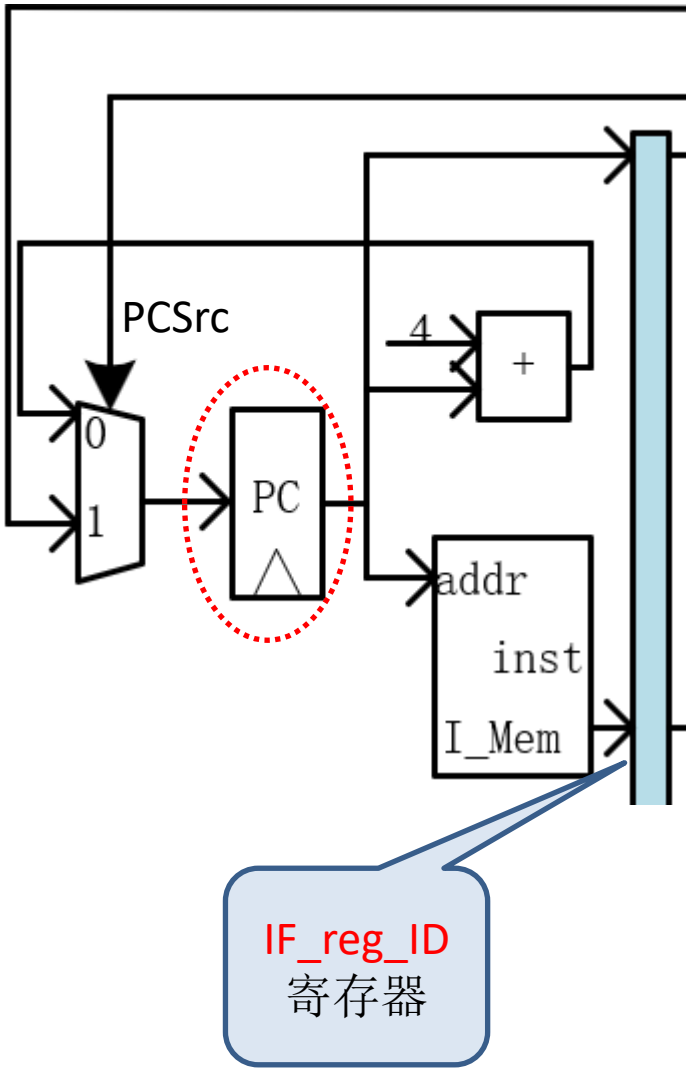
取指—功能介绍 Instruction Fetch (IF)



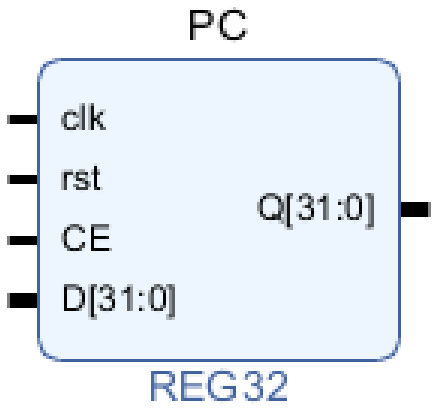
- 取指：取指阶段涉及程序计数器（PC）和指令存储器（I_Mem）；程序计数器输出作为地址从指令存储器中读取指令。
- IF_reg_ID：暂存指令和PC值，以待下一级使用

取指—部件介绍

- 程序计数器（PC）



- 程序计数器（PC）：本质是一个32位的寄存器，用于锁存地址，其输出作为访存地址读指令存储器，访存结果为当前指令内容。



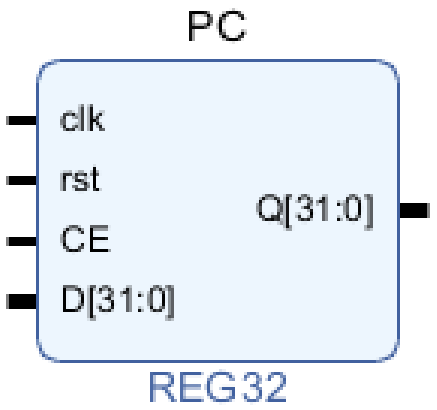
取指—部件介绍

- 程序计数器（PC）

- 模块名：REG32
  - 上升沿触发：clk
  - 使能信号：CE
  - 同步复位：rst=1
  - 数据输入：D(31:0)
  - 数据输出：Q(31:0)
- 参考描述结构

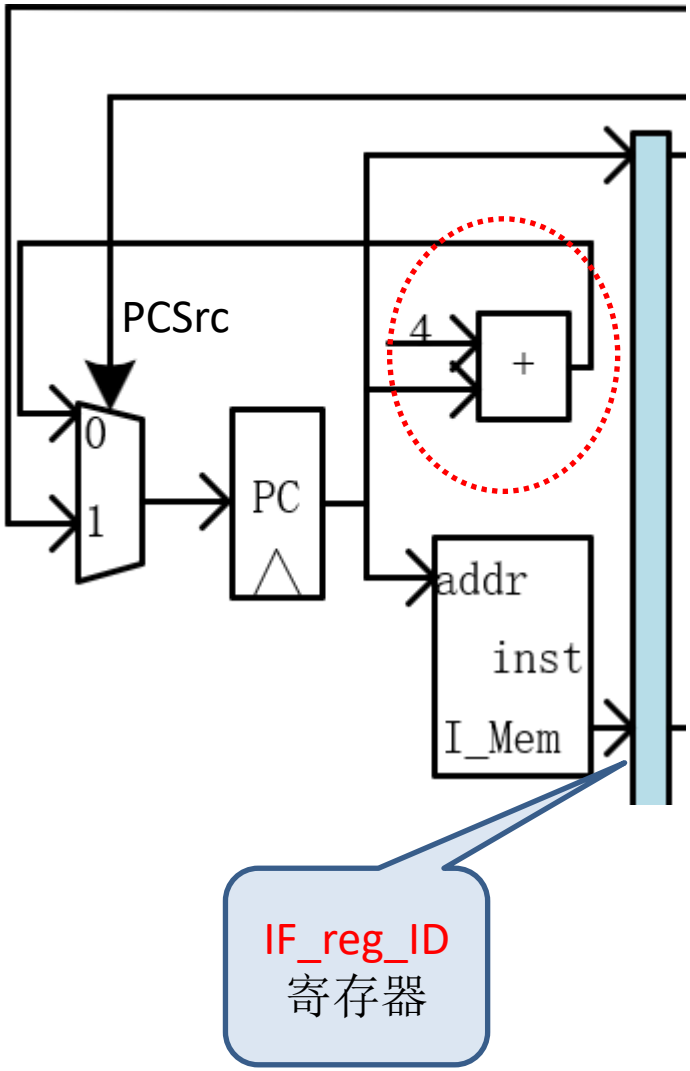
```
module   REG32(input clk,
                ……

      always @(posedge clk or posedge rst)
                if (rst==？ )  Q <= ？ ;
                else if (？ )   Q <= ？ ;
endmodule
```

可直接调用OExp04的PC寄存器模块

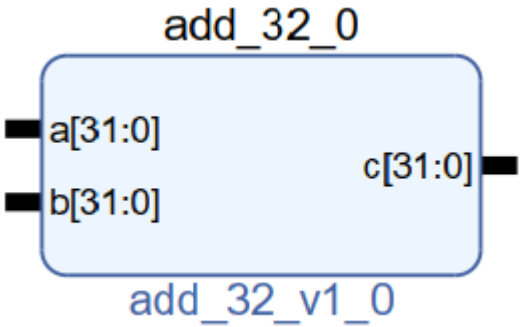


取指—部件介绍

- 加法器（add）



- 加法器（add）：本质是一个二输入32位的加法器，用于计算PC+4形成顺序执行程序时新的PC值。



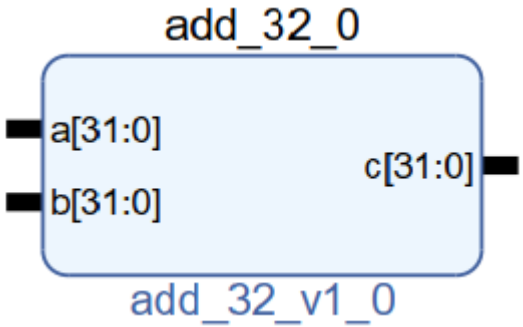
取指—部件介绍

- 加法器（add）

- 模块名：add_32
  - 数据输入：a(31:0)
  - 数据输入：b(31:0)
  - 数据输出：c(31:0)
- 参考描述结构

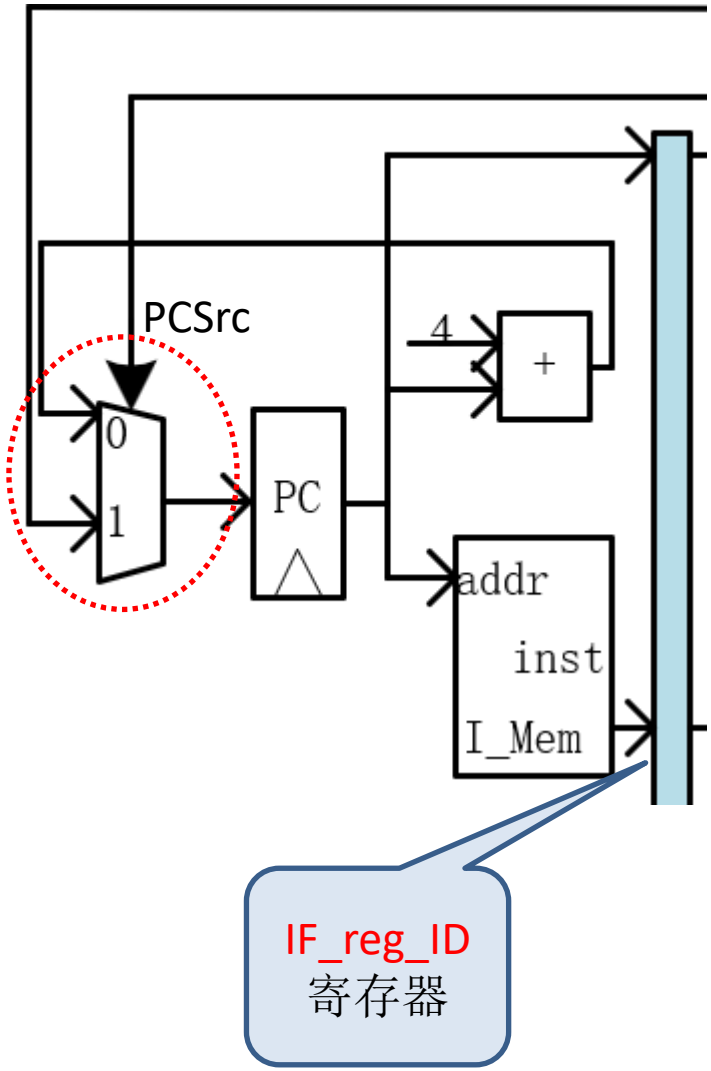
```
module  add_32(input [31:0] a,
               input [31:0] b,
               output [31:0] c
              );

              ……

endmodule
```

可直接调用OExp01的加法器模块

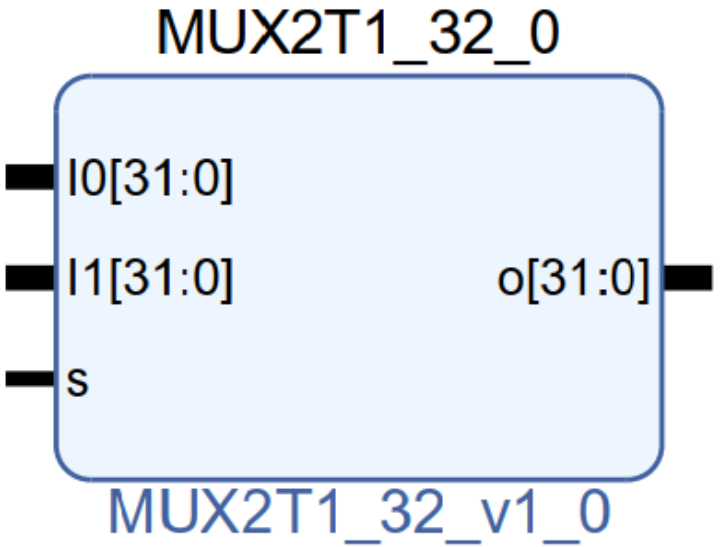


取指—部件介绍

- 多选器（MUX2T1_32）



- 多选器（MUX2T1_32）：本质是一个32位二选一的多路器，用于选择PC的更新值，当PCSrc=0;选择PC+4,当PCSrc=1;选择跳转目标地址。



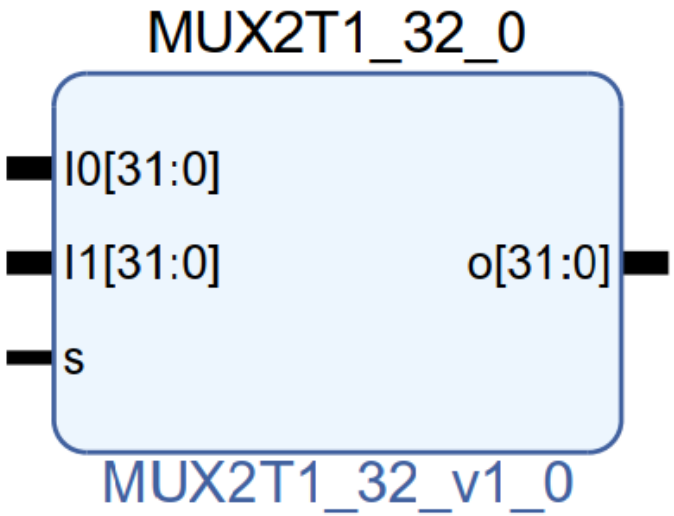
取指—部件介绍

- 多选器（MUX2T1_32）

- 模块名：MUX2T1_32
  - 数据输入：I0(31:0)
  - 数据输入：I1(31:0)
  - 数据输入：S
  - 数据输出：O(31:0)
- 参考描述结构

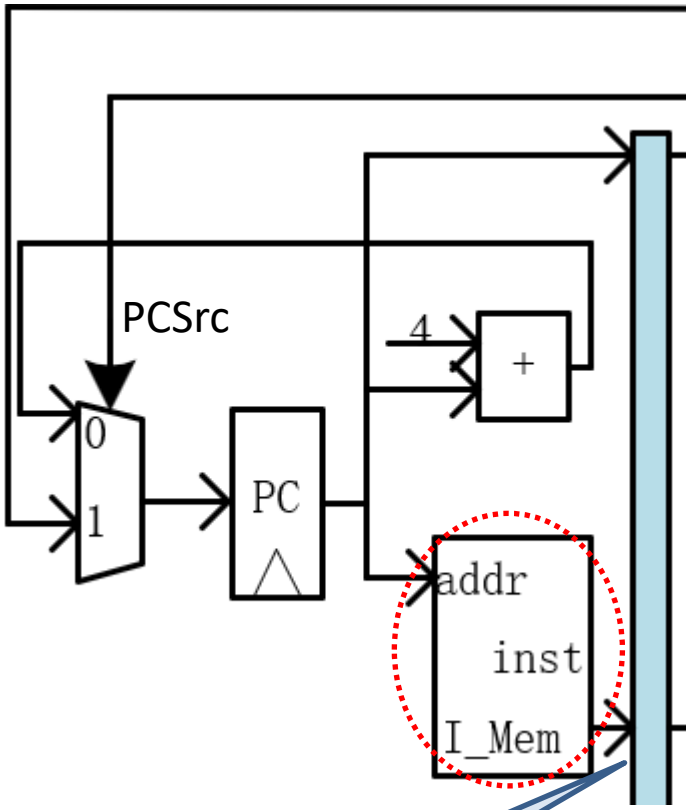
```
module   MUX2T1_32(input [31:0] I0,
                   input [31:0] I1,
                   input sel,
                   output reg[31:0] o
                   );
          ……
endmodule
```

可直接调用OExp01的多选器模块



取指—部件介绍

- 指令存储器（I_Mem）



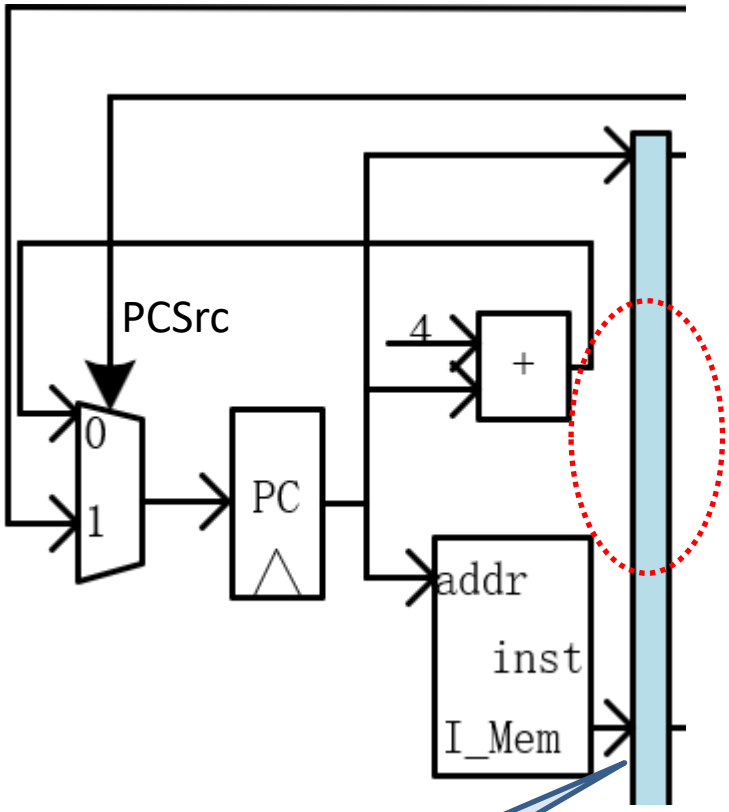
- 指令存储器：本质是一个异步ROM，输入地址则输出数据，用于存储CPU的指令（I_Mem不是CPU的内部部件，只在构建SOC系统时才用到）。

可直接仿照OExp01由IP工具生成

IF_reg_ID寄存器

取指—部件介绍

- 取指-译码寄存器（IF_reg_ID）



- 取指-译码寄存器（IF_reg_ID）：本质是一个寄存器，用于寄存PC值和指令存储器输出的指令。

本实验设计

IF_reg_ID
寄存器

取指—部件介绍

■ 取指-译码寄存器（IF_reg_ID）

◎ **IF_reg_ID**

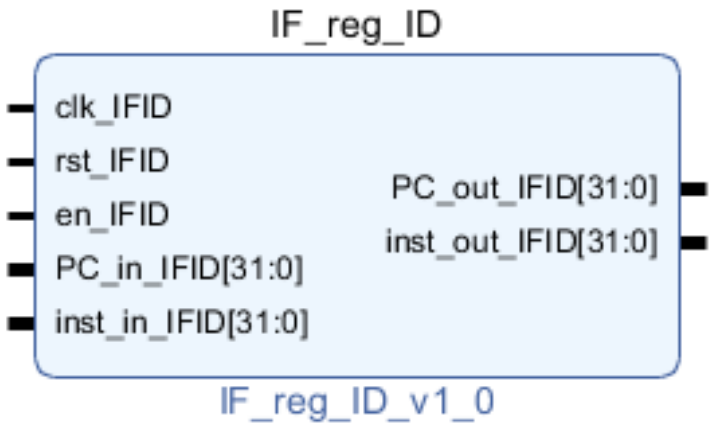
- 流水线CPU取指和译码之间的寄存器
- 存储PC值和指令

◎ **基本功能**

- 寄存IF级的输出指令，分割IF级和ID级的指令或控制信号，防止相互干扰，在IF级执行结束时将指令的控制信号传递至下一级。

◎ **接口要求**

- 取指译码寄存器接口如图：



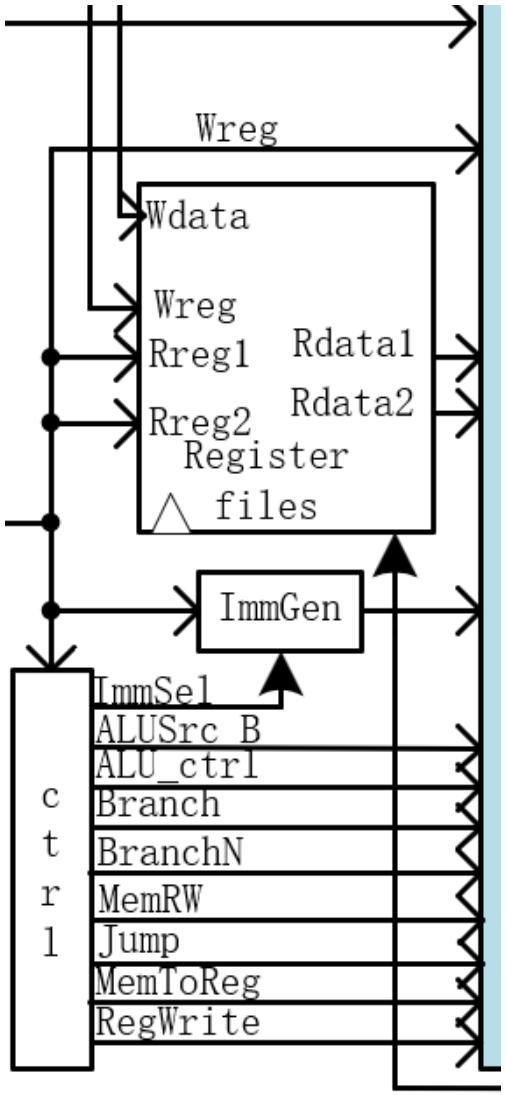
取指-译码寄存器接口：IF_reg_ID.v

```
module    IF_reg_ID(
            input              clk_IFID,              //寄存器时钟
            input              rst_IFID,              //寄存器复位
            input              en_IFID,               //寄存器使能
            input  [31:0]      PC_in_IFID,            //PC输入
            input  [31:0]      inst_in_IFID,          //指令输入

            output reg [31:0]  PC_out_IFID，          //PC输出
            output reg [31:0]  inst_out_IFID          //指令输出
                        );
    always @(posedge clk_IFID or posedge rst_IFID)
                    if (rst_IFID==？ ) …… <= ？ ;
                    else if (？ )  …… <= ？ ;

endmodule
```

译码—功能介绍

- Instruction Decoder(ID

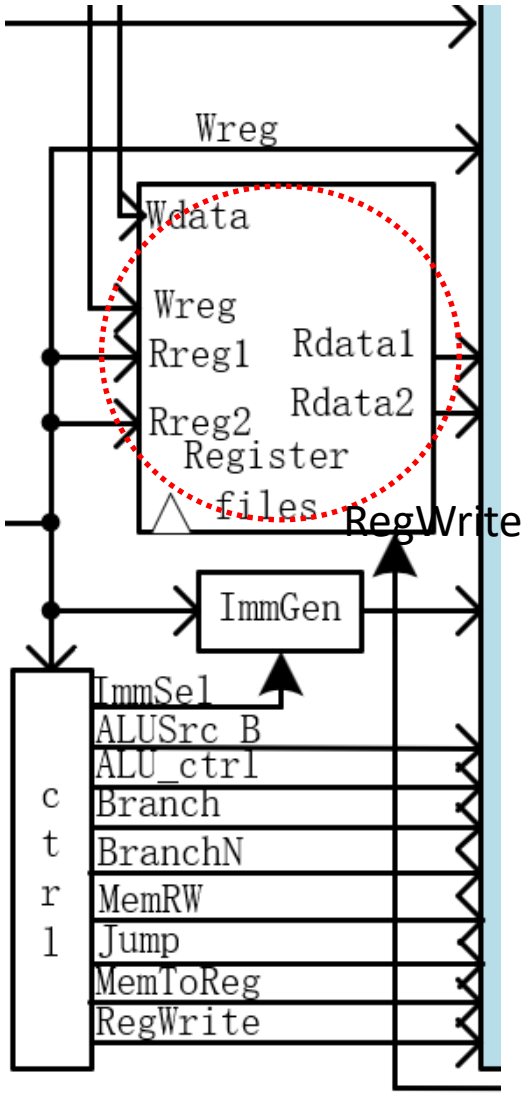
译码器即为单周期CPU中的控制器



- 译码：译码阶段涉及寄存器堆（RegisterFiles）和译码器、立即数生成单元（ImmGen）；从寄存器堆可以读取操作数，译码器对指令进行解析产生各种各种控制信号，立即数生成单元根据控制信号和输入指令生成各种类型的立即数。
- ID_reg_Ex：暂存PC值，寄存器读取数据，立即数和控制信号以待下一级使用

ID_reg_Ex
寄存器

译码—部件介绍 ■ 寄存器堆(RegFiles)



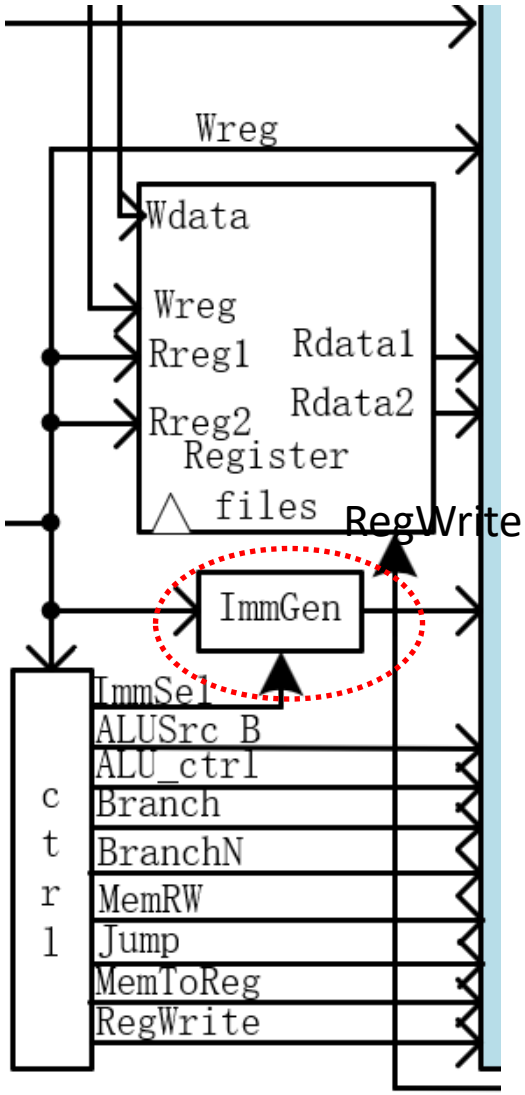
- 寄存器堆：本质是32×32bit寄存器组，指令操作的主要对象；根据寄存器号读取和写入数据，带有写控制信号。

可直接调用OExp04寄存器堆模块

ID_reg_Ex 寄存器

译码—部件介绍

- 立即数生成单元(ImmGen)



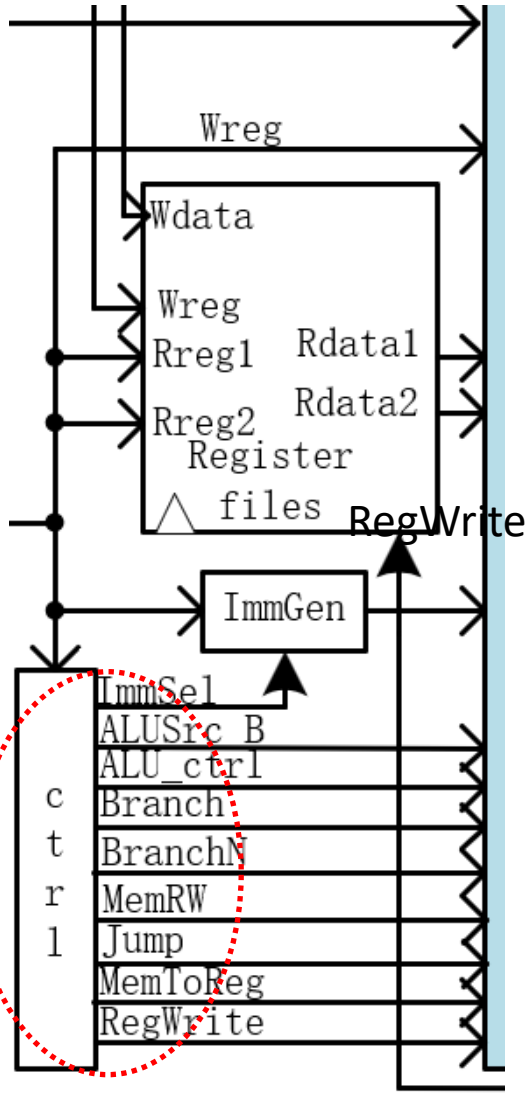
- 立即数生成单元：根据输入指令生成立即数。

可直接调用OExp04立即数生成模块

ID_reg_Ex寄存器

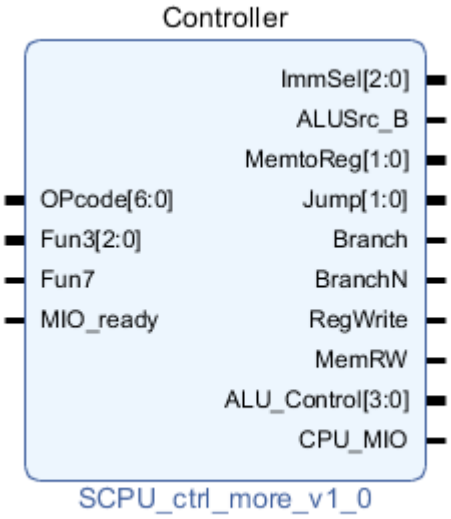
译码—部件介绍

- 译码模块/控制器



- 译码模块/控制器：根据输入指令译码输出各个部件的控制信号。

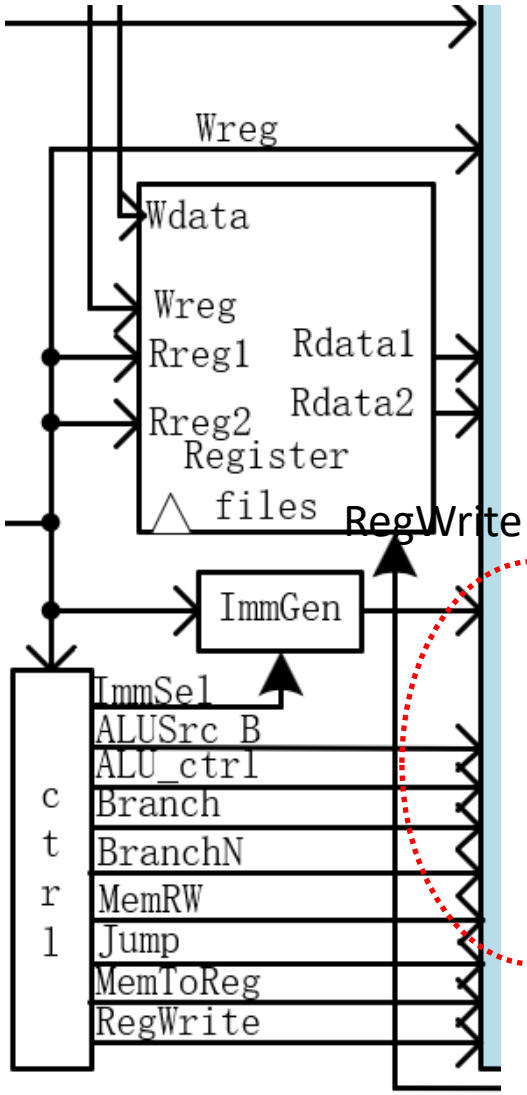
可直接调用OExp06控制器模块



ID_reg_Ex
寄存器

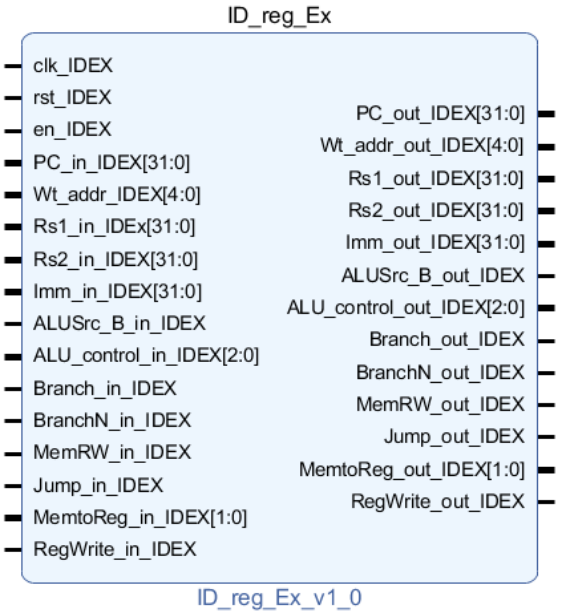
译码—部件介绍

- 译码-执行寄存器（ID_reg_Ex）



- 译码-执行寄存器：寄存控制信号和ID输出数据。

本实验设计



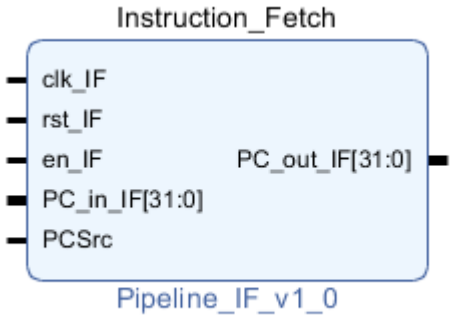
ID_reg_Ex
寄存器

- 任务一：设计取指（IF）、译码（ID）模块，替换实验九的流水线CPU并完成集成
  - 设计取指模块，替换OExp05-1的取指模块并完成集成
  - 设计译码模块，替换OExp05-1的译码模块并完成集成

取指模块、IF_reg_ID寄存器设计集成

Pipeline—IF　　取指模块接口

- **Pipeline_IF**
  - 流水线CPU第一阶段
  - 根据程序计数器从指令存储器中取出指令
- **基本功能**
  - 由程序计数器获取PC值及PC值的更新
  - 由PC值获取指令
- **接口要求**
  - 取指模块接口如图：

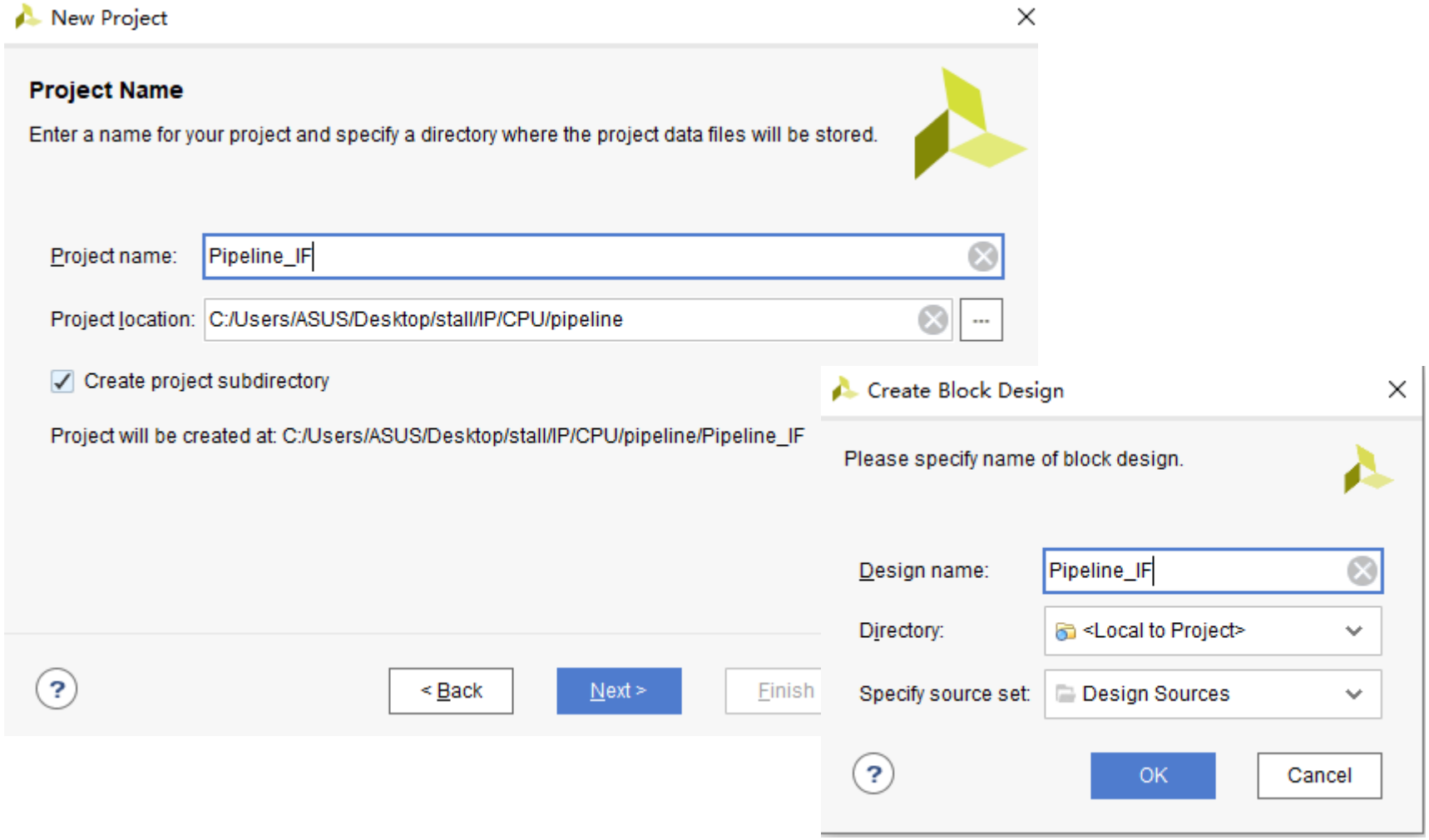


取指模块接口信号标准：Pipeline_IF.v

```
module     Pipeline_IF(
             input                clk_IF,          //时钟
             input                rst_IF,          //复位
             input                en_IF,           //使能
             input        [31:0]  PC_in_IF,        //取指令PC输入
             input                PCSrc,           //PC输入选择
             output reg [31:0]    PC_out_IF
                          );
endmodule
```

（PCSrc=(Branch&zero)|(BranchN&(~zero))|Jump）

取指模块设计


New Project
Project Name
Enter a name for your project and specify a directory where the project data files will be stored.
Project name: Pipeline_IF
Project location: C:/Users/ASUS/Desktop/stall/IP/CPU/pipeline
Create project subdirectory
Project will be created at: C:/Users/ASUS/Desktop/stall/IP/CPU/pipeline/Pipeline_IF
< Back
Next >
Finish
Create Block Design
Please specify name of block design.
Design name: Pipeline_IF
Directory: <Local to Project>
Specify source set: Design Sources
OK
Cancel

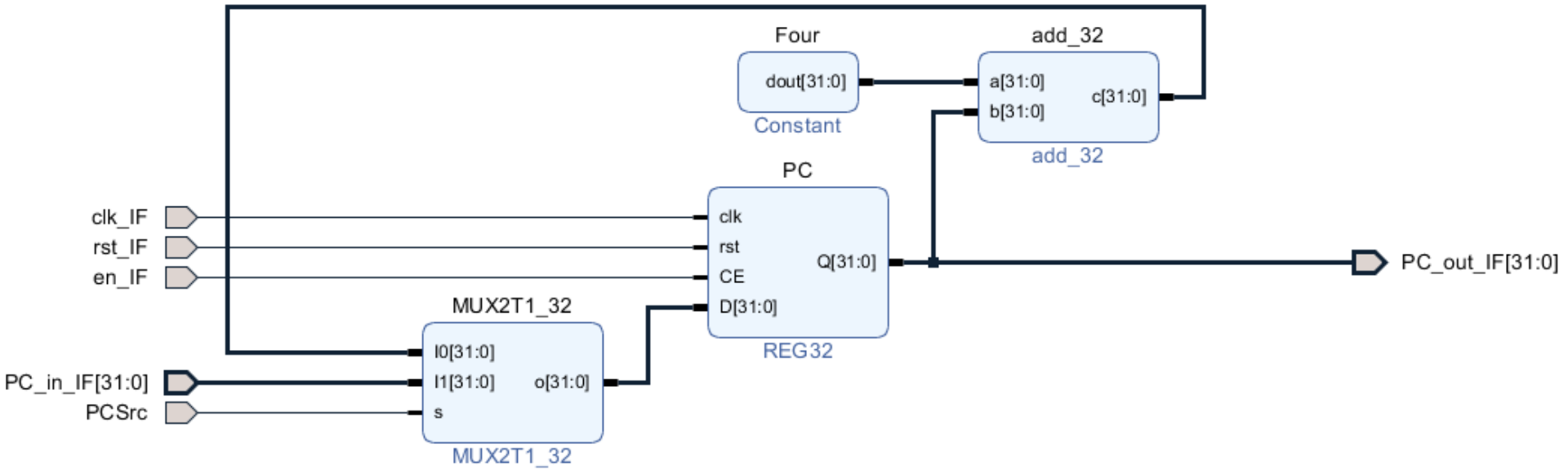
取指模块设计

拷贝下列模块到Pipeline_IF工程目录：

MUX2T1_32、REG32、add_32

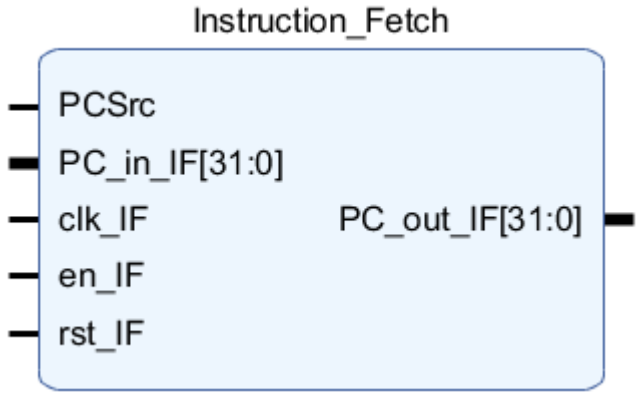
添加模块路径到Pipeline_IF工程目录：

取指模块设计


Four
dout[31:0]
Constant
add_32
a[31:0]
b[31:0]
c[31:0]
add_32
PC
clk
rst
CE
D[31:0]
Q[31:0]
REG32
clk_IF
rst_IF
en_IF
PC_out_IF[31:0]
MUX2T1_32
I0[31:0]
I1[31:0]
s
o[31:0]
PC_in_IF[31:0]
PCSrc
MUX2T1_32

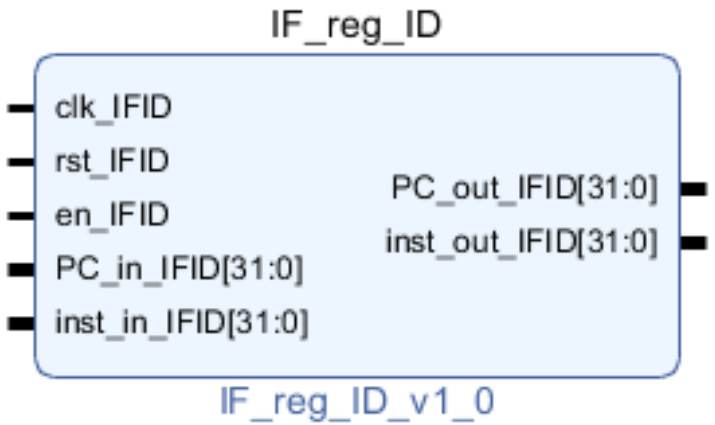
取指模块设计

- Pipeline_IF (Pipeline_IF.v) (4)
  - Four : Pipeline_IF_xlconstant_0_0 (Pipeline_IF_xlconstant_0_0.xci)
  - MUX2T1_32 : Pipeline_IF_MUX2T1_32_0_0 (Pipeline_IF_MUX2T1_32_0_0.xci)
  - PC : Pipeline_IF_REG32_0_0 (Pipeline_IF_REG32_0_0.xci)
  - add_32 : Pipeline_IF_add_32_0_0 (Pipeline_IF_add_32_0_0.xci)



Pipeline—IF_reg_ID　　取指-译码寄存器接口

- **IF_reg_ID**
  - 流水线CPU取指和译码之间的寄存器
  - 存储PC值和指令
- **基本功能**
  - 寄存IF级的输出指令，分割IF级和ID级的指令或控制信号，防止相互干扰，在IF级执行结束时将指令的控制信号传递至下一级。
- **接口要求**
  - 取指译码寄存器接口如图：



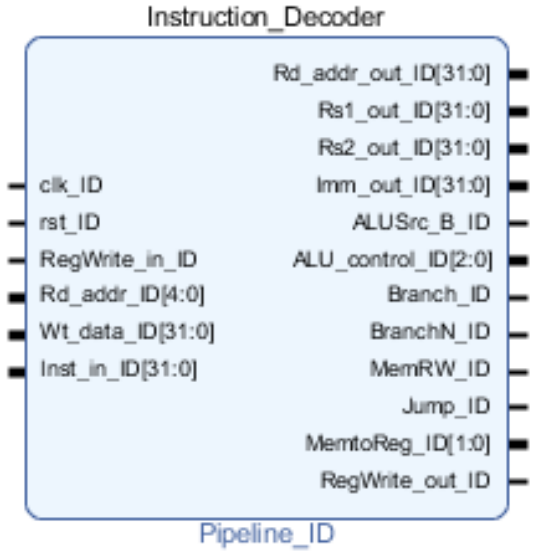
取指-译码寄存器接口：IF_reg_ID.v

```
module    IF_reg_ID(
            input              clk_IFID,            //寄存器时钟
            input              rst_IFID,            //寄存器复位
            input              en_IFID,             //寄存器使能
            input  [31:0]      PC_in_IFID,          //PC输入
            input  [31:0]      Inst_in_IFID,        //指令输入

            output reg [31:0]  PC_out_IFID，        //PC输出
            output reg [31:0]  inst_out_IFID        //指令输出

                        );
endmodule
```

译码模块、ID_reg_Ex寄存器设计集成

Pipeline—ID　译码模块接口

- **Pipeline_ID**
  - 流水线CPU第二阶段
  - 指令译码
- **基本功能**
  - 译码是指将从指令存储器取指的指令进行翻译的过程；译码之后产生各种控制信号，同时寄存器堆根据所需操作数寄存器的索引读出操作数，立即数生成单元输出所需立即数。
- **接口要求**
  - 译码模块接口如图：



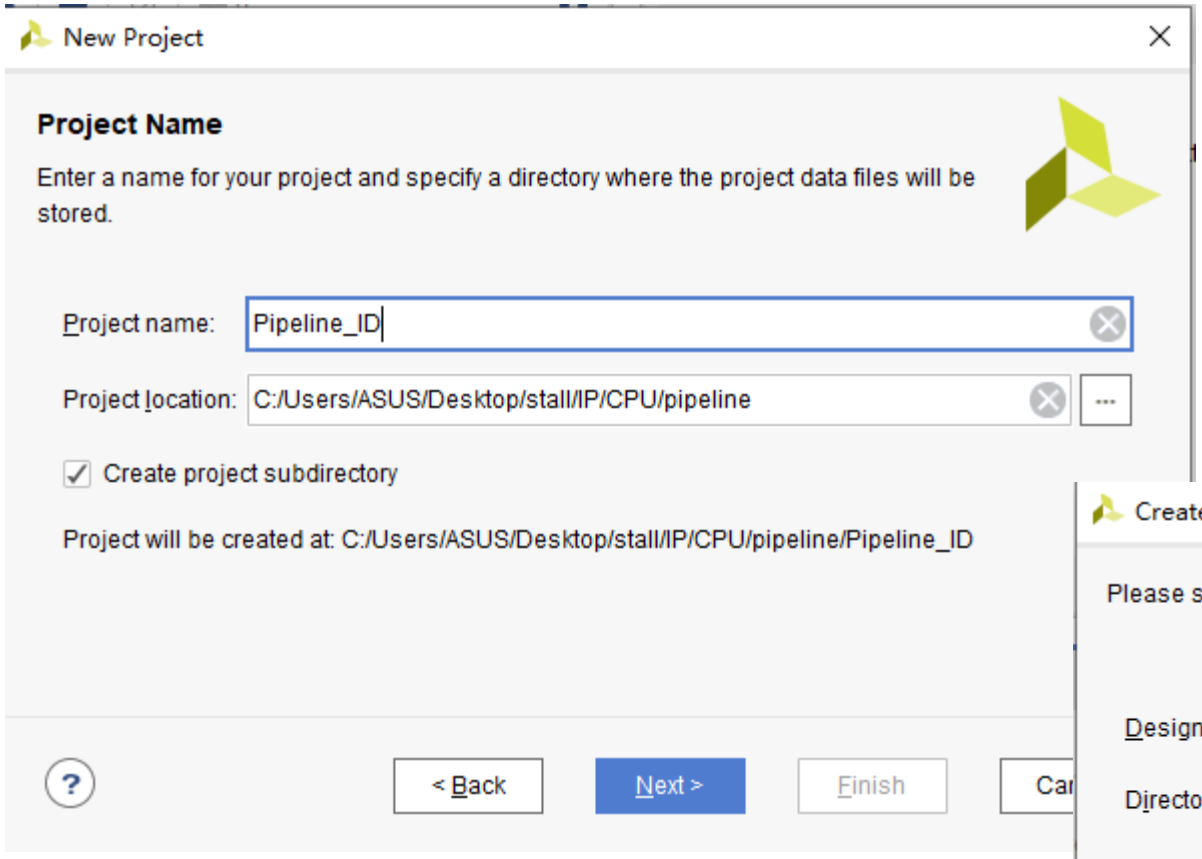
译码模块接口：Pipeline_ID.v

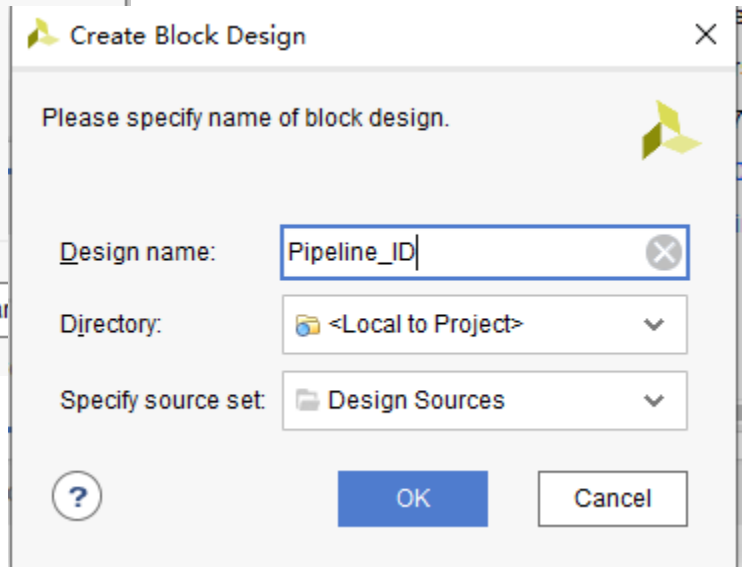
```
module   Pipeline_ID(
          input            clk_ID,              //时钟
          input            rst_ID,              //复位
          input            RegWrite _in_ID,     //寄存器堆使能
          input  [4:0]     Rd_addr_ID,          //写目的地址输入
          input  [31:0]    Wt_data_ID,          //写数据输入
          input  [31:0]    Inst_in_ID,          //指令输入
                 ……
```

译码模块接口：Pipeline_ID.v

```
.............
output reg [31:0]   Rd_addr_out_ID, //写目的地址输出
output reg [31:0]   Rs1_out_ID,     //操作数1输出
output reg [31:0]   Rs2_out_ID ,    //操作数2输出
output reg [31:0]   Imm_out_ID ,    //立即数输出
output reg          ALUSrc_B_ID ,   //ALU B端输入选择
output reg [2:0]    ALU_control_ID, //ALU控制
output reg          Branch_ID,      //Beq控制
output reg          BranchN_ID,     //Bne控制
output reg          MemRW_ID,       //存储器读写
output reg          Jump_ID,        //Jal控制
output reg [1:0]    MemtoReg_ID,    //寄存器写回选择
output reg          RegWrite_out_ID, //寄存器堆读写
                );
endmodule
```

译码模块设计


New Project
Project Name
Enter a name for your project and specify a directory where the project data files will be stored.
Project name: Pipeline_ID
Project location: C:/Users/ASUS/Desktop/stall/IP/CPU/pipeline
Create project subdirectory
Project will be created at: C:/Users/ASUS/Desktop/stall/IP/CPU/pipeline/Pipeline_ID
< Back
Next >
Finish


Create Block Design
Please specify name of block design.
Design name: Pipeline_ID
Directory: <Local to Project>
Specify source set: Design Sources
OK
Cancel

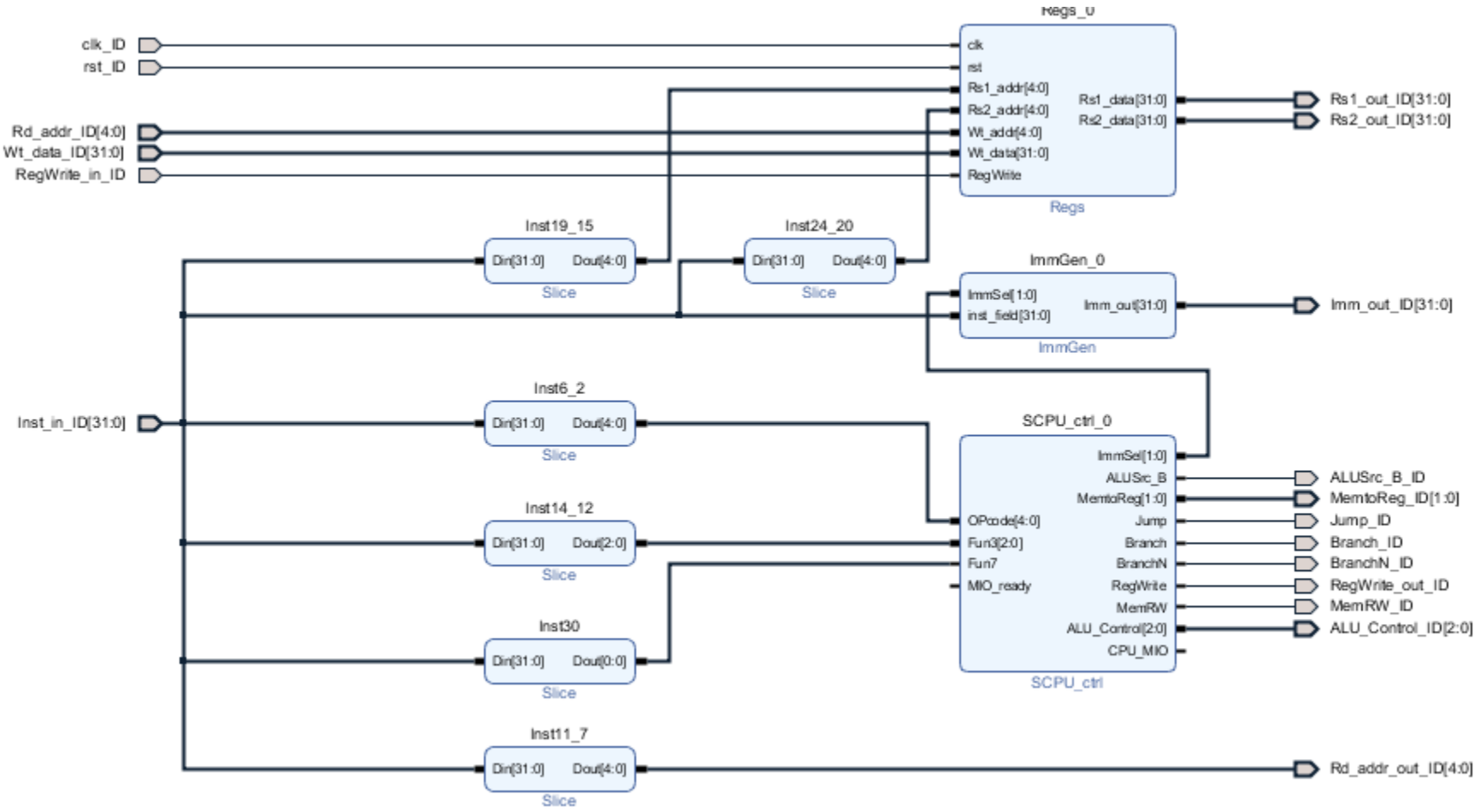
译码模块设计

拷贝下列模块到Instruction_Decoder工程目录：

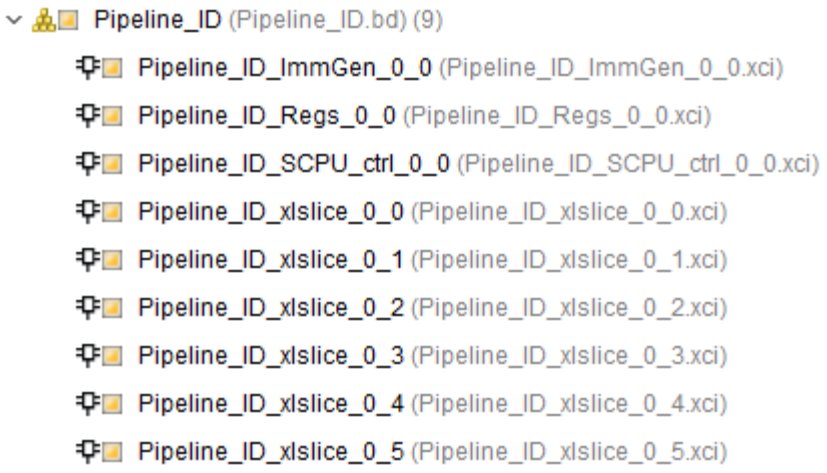
ImmGen、Regs、SCPU_Ctrl

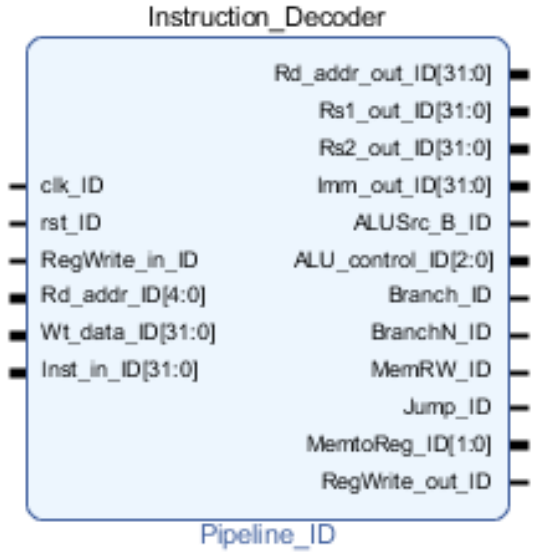
添加模块路径到Instruction_Decoder工程目录：

译码模块设计


clk_ID
rst_ID
Rd_addr_ID[4:0]
Wt_data_ID[31:0]
RegWrite_in_ID
Inst_in_ID[31:0]
Regs_0
clk
rst
Rs1_addr[4:0]
Rs2_addr[4:0]
Wt_addr[4:0]
Wt_data[31:0]
RegWrite
Rs1_data[31:0]
Rs2_data[31:0]
Regs
Rs1_out_ID[31:0]
Rs2_out_ID[31:0]
Inst19_15
Din[31:0]
Dout[4:0]
Slice
Inst24_20
Din[31:0]
Dout[4:0]
Slice
ImmGen_0
ImmSel[1:0]
inst_field[31:0]
Imm_out[31:0]
ImmGen
Imm_out_ID[31:0]
Inst6_2
Din[31:0]
Dout[4:0]
Slice
SCPU_ctrl_0
ImmSel[1:0]
ALUSrc_B
MemtoReg[1:0]
OPcode[4:0]
Jump
Fun3[2:0]
Branch
Fun7
BranchN
MIO_ready
RegWrite
MemRW
ALU_Control[2:0]
CPU_MIO
SCPU_ctrl
ALUSrc_B_ID
MemtoReg_ID[1:0]
Jump_ID
Branch_ID
BranchN_ID
RegWrite_out_ID
MemRW_ID
ALU_Control_ID[2:0]
Inst14_12
Din[31:0]
Dout[2:0]
Slice
Inst30
Din[31:0]
Dout[0:0]
Slice
Inst11_7
Din[31:0]
Dout[4:0]
Slice
Rd_addr_out_ID[4:0]

译码模块设计

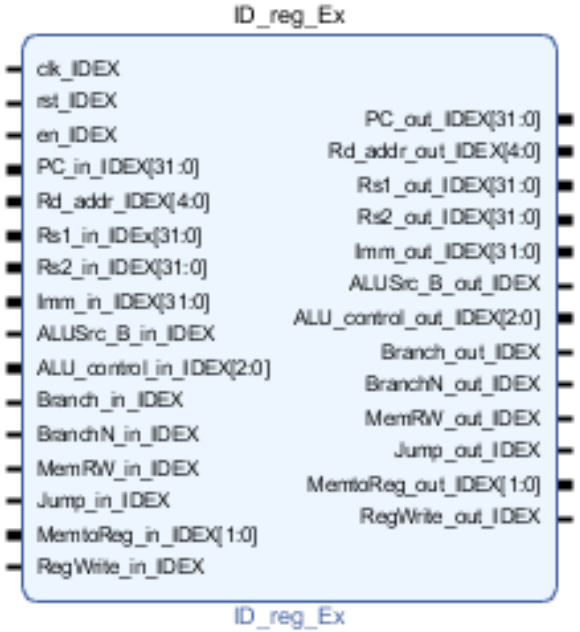




根据原理图采用RTL实现

Pipeline—ID_reg_Ex　　译码-执行寄存器接口

- **ID_reg_Ex**
  - 流水线CPU译码和执行之间的寄存器
  - 存储ALU数据和控制信号
- **基本功能**
  - 寄存ID级的输出指令，分割ID级和EX级的指令或控制信号，防止相互干扰，在ID级执行结束时将指令的控制信号传递至下一级。。
- **接口要求**
  - 译码执行寄存器接口如图：



译码-执行寄存器接口：ID_reg_Ex.v

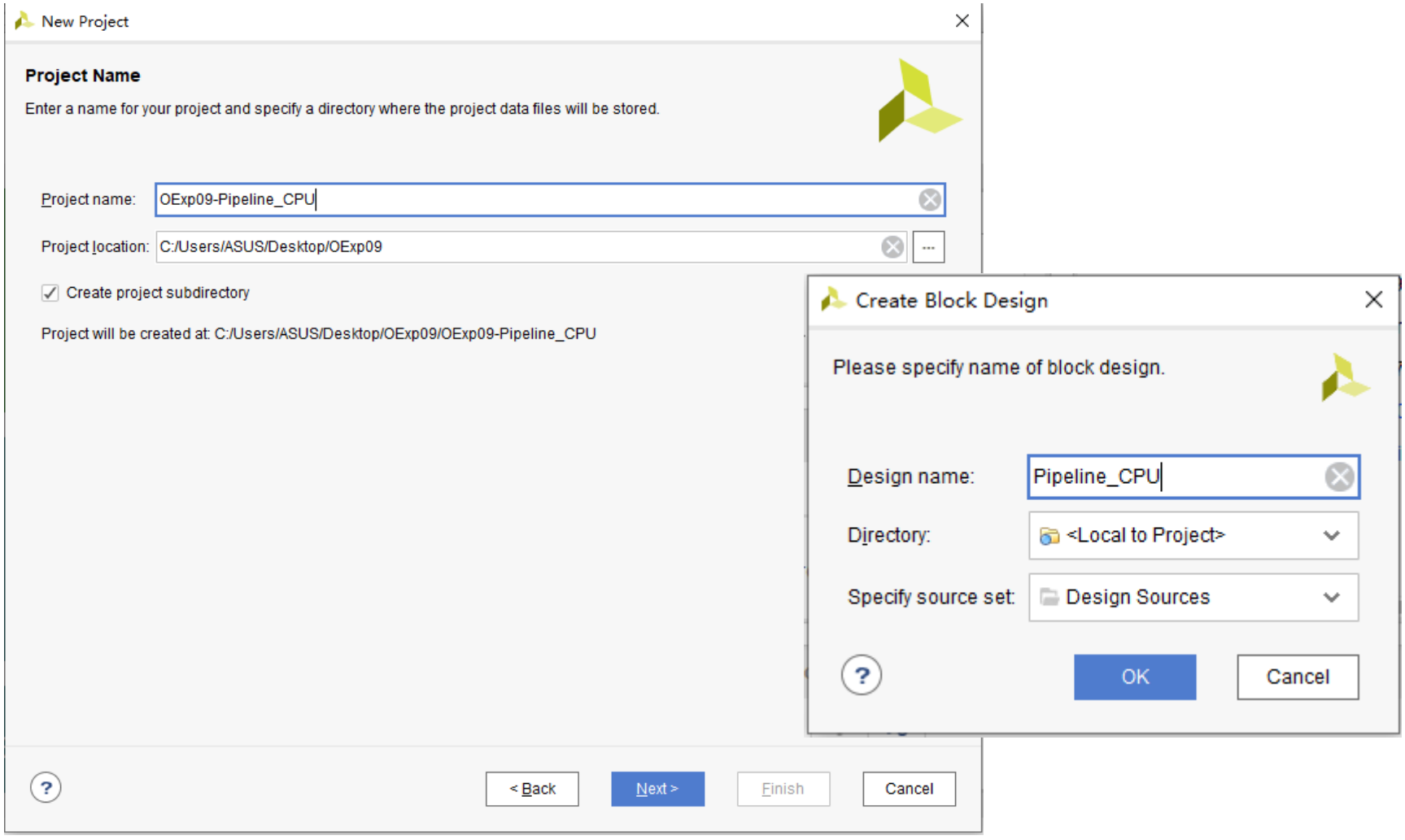
```
module    ID_reg_Ex(
           input          clk_IDEX,                //寄存器时钟
           input          rst_IDEX,                //寄存器复位
           input          en_IDEX,                 //寄存器使能
           input[31:0]    PC_in_IDEX,              //PC输入
           input[4:0]     Rd_addr_IDEX,            //写目的地址输入
           input[31:0]    Rs1_in_IDEX，            //操作数1输入
           input[31:0]    Rs2_in_ID EX,            //操作数2输如
           input[31:0]    Imm_in_IDEX ,            //立即数输入
           input          ALUSrc_B_in_IDEX ,       //ALU B输入选择
           input[2:0]     ALU_control_in_IDEX,     //ALU选择控制
           input          Branch_in_IDEX,          //Beq
           input          BranchN_in_IDEX,         //Bne
           input          MemRW_in_IDEX,           //存储器读写
           input          Jump_in_IDEX,            //Jal
           input[1:0]     MemtoReg_in_IDEX,        //写回选择
           input          RegWrite_in_IDEX,        //寄存器堆读写
           ……
```

译码-执行寄存器接口：ID_reg_Ex.v

```
.....................................
                output reg[31:0]   PC_out_IDEX，              //PC输出
                output reg[4:0]    Rd_addr_out_IDEX           //目的地址输出
                output reg[31:0]   Rs1_out_IDEX，             //操作数1输出
                output reg[31:0]   Rs2_out_ID EX,             //操作数2输出
                output reg[31:0]   Imm_out_IDEX ,             //立即数输出
                output reg         ALUSrc_B_out_IDEX ,        //ALU B选择
                output reg[2:0]    ALU_control_out_IDEX,      //ALU控制
                output reg         Branch_out_IDEX,           //Beq
                output reg         BranchN_out_IDEX,          //Bne
                output reg         MemRW_out_IDEX,            //存储器读写
                output reg         Jump_out_IDEX,             //Jal
                output reg [1:0]   MemtoReg_out_IDEX,         //写回
                output reg         RegWrite_out_IDEX          //寄存器堆读写
                   );
endmodule
```

IF、ID模块替换与流水线CPU集成

流水线CPU集成


New Project
Project Name
Enter a name for your project and specify a directory where the project data files will be stored.
Project name:
OExp09-Pipeline_CPU
Project location:
C:/Users/ASUS/Desktop/OExp09
Create project subdirectory
Project will be created at: C:/Users/ASUS/Desktop/OExp09/OExp09-Pipeline_CPU
< Back
Next >
Finish
Cancel
Create Block Design
Please specify name of block design.
Design name:
Pipeline_CPU
Directory:
<Local to Project>
Specify source set:
Design Sources
OK
Cancel

清理Exp05-1工程

- 移除工程中的取指、译码模块
- 建议用Exp05-1资源重建工程

设计要点

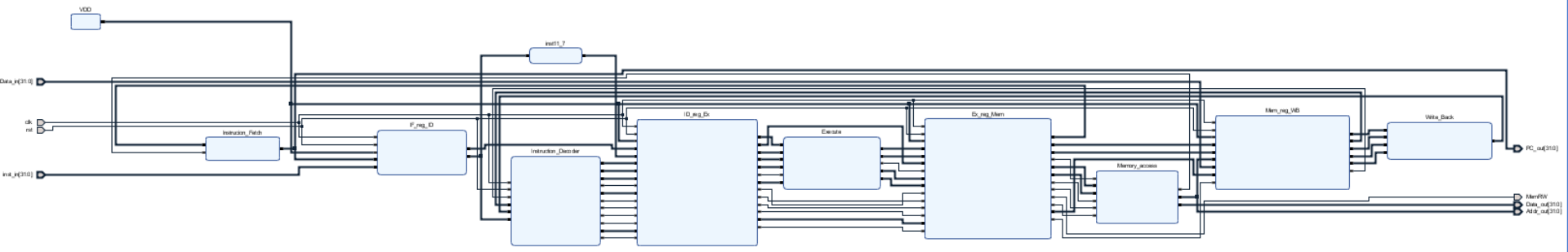
本实验设计

拷贝下列模块到Pipeline_CPU工程目录：

Pipeline_IF、IF_reg_ID、Pipeline_ID、ID_reg_Ex、

添加模块路径到Pipeline_CPU工程目录：

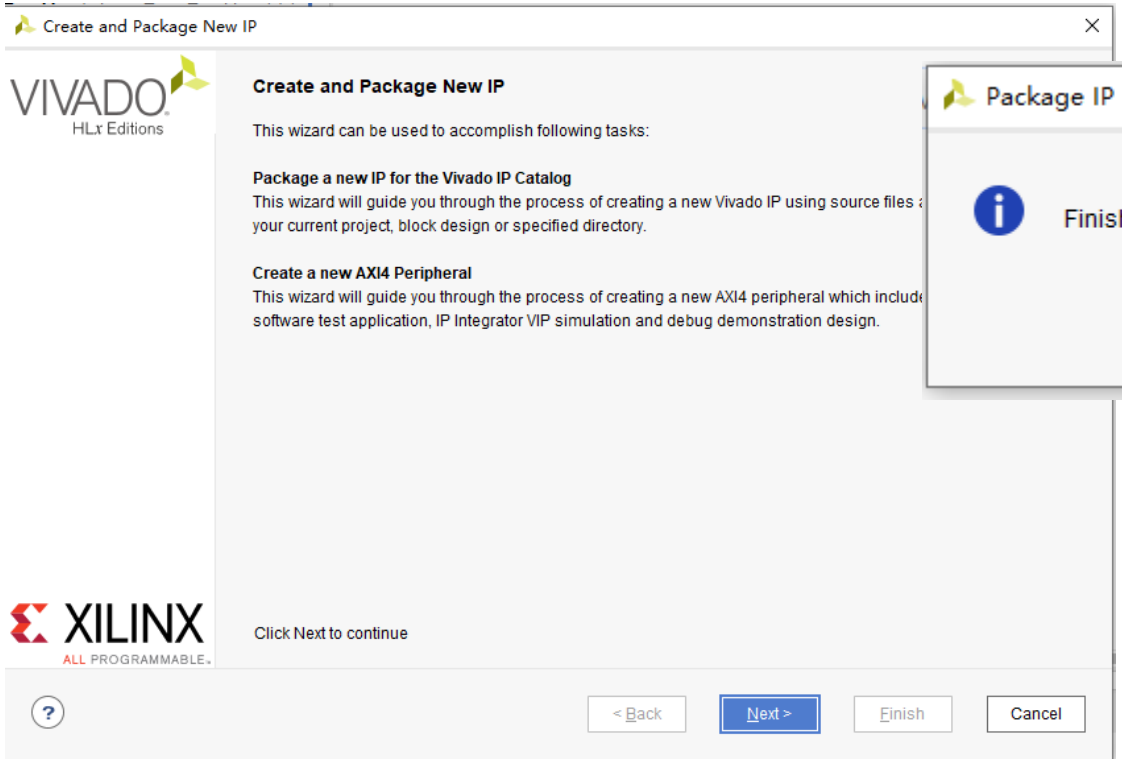
流水线CPU集成----替换取指、译码模块

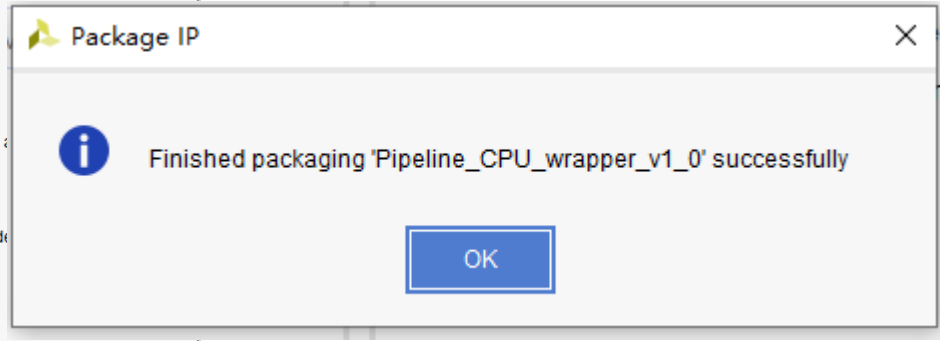


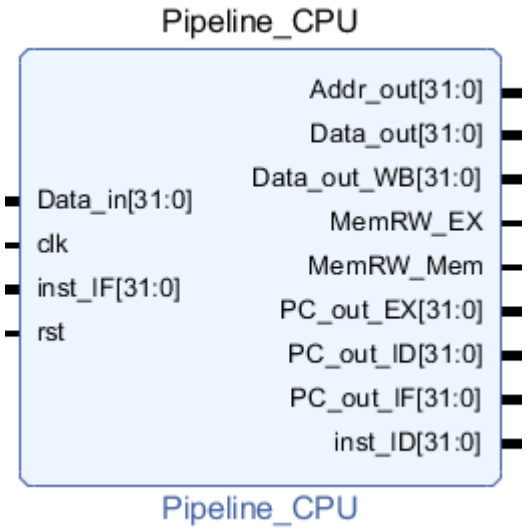
集成Pipeline CPU的模块层次结构

五级模块、四个寄存器

流水线CPU集成


Create and Package New IP
VIVADO
HLx Editions
Create and Package New IP
This wizard can be used to accomplish following tasks:
Package a new IP for the Vivado IP Catalog
This wizard will guide you through the process of creating a new Vivado IP using source files
your current project, block design or specified directory.
Create a new AXI4 Peripheral
This wizard will guide you through the process of creating a new AXI4 peripheral which include
software test application, IP Integrator VIP simulation and debug demonstration design.
XILINX
ALL PROGRAMMABLE
Click Next to continue
< Back
Next >
Finish
Cancel


Package IP
Finished packaging 'Pipeline_CPU_wrapper_v1_0' successfully
OK


Pipeline_CPU
Data_in[31:0]
clk
inst_IF[31:0]
rst
Addr_out[31:0]
Data_out[31:0]
Data_out_WB[31:0]
MemRW_EX
MemRW_Mem
PC_out_EX[31:0]
PC_out_ID[31:0]
PC_out_IF[31:0]
inst_ID[31:0]
Pipeline_CPU

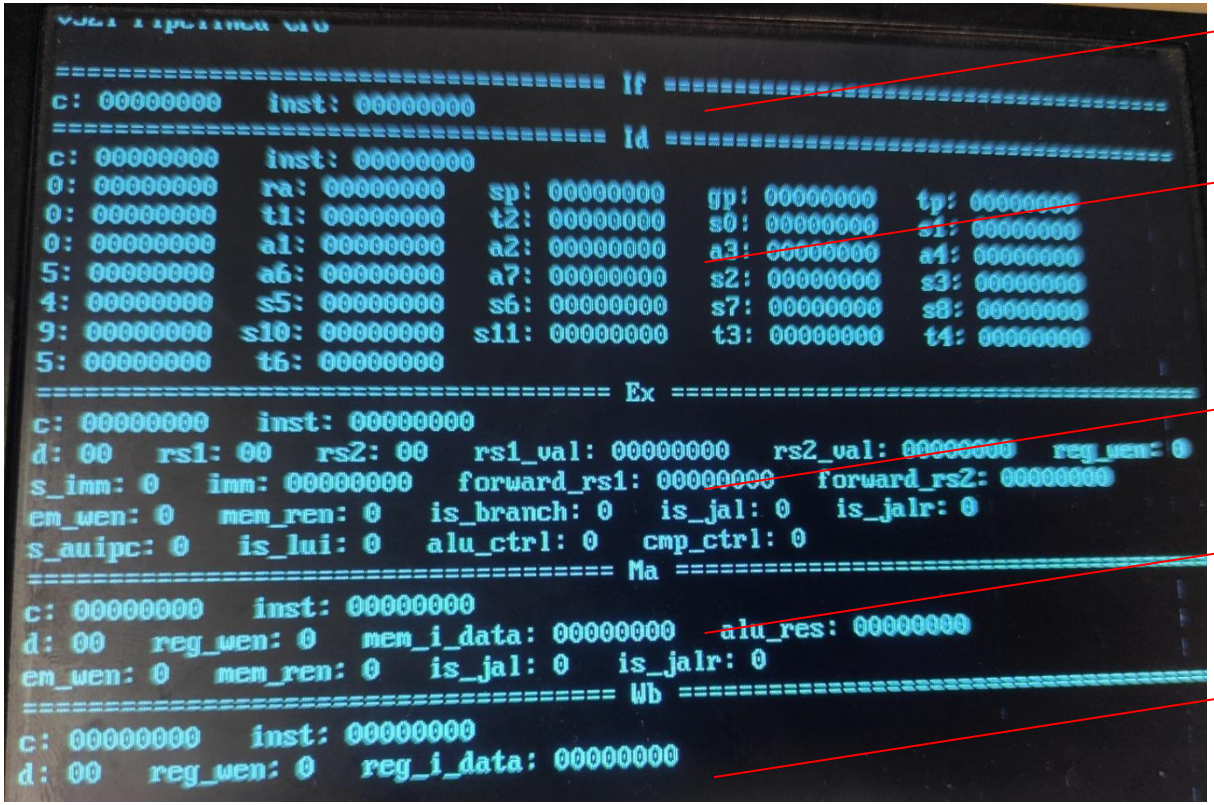
- 任务二：设计流水线测试方案并完成测试

物理验证

- **使用DEMO程序目测CPU运行情况**
  - DEMO接口功能
    - SW[8]=0， SW[2]=0(全速运行)
    - SW[8]=0， SW[2]=1(自动单步)
    - SW[8]=1， SW[2]=x(手动单步)
- **用汇编语言设计测试程序**
  - 测试ALU指令(R-格式译码\I-立即数格式译码)
  - 测试LW指令(I-格式译码)
  - 测试SW指令(S-格式译码)
  - 测试分支指令(B-格式译码)

物理验证

- 为更好追踪流水线CPU的特点，VGA显示的接口稍有调整，分别从取指、译码、执行、访存、写回进行显示，请采用更新版本的IP
- 实验中选取了部分信号进行观测，若想观察其他信号，请参照Lab04将其他待测信号引出即可



取指阶段信号

译码阶段信号

执行阶段信号

访存阶段信号

写回阶段信号

测试程序参考：

- 方案一： 无冒险的流水线测试（p.mem）

```
#baseAddr 0000
main:   addi x1,x0,0x1        #x1 = 0x1
        addi x2,x0,0x1        #x2 = 0x1
        addi x3,x0,0x1        #x3 = 0x1
        addi x4,x0,0x1        #x4 = 0x1
        lw x5,0x8(x0)      #x5 = 0x80000000
        add x6,x1,x1          #x6 = 0x2
        xor x7,x1,x2          #x7 = 0
        sub x8,x2,x1          #x8 = 0
        ori x9,x3,-1        #x9 = 0xFFFFFFFF
        and x10,x4,x3         #x10= 0x2
        sw x5,0x4(x0)         #mem(1)=
                              0x80000000
        slt x11,x6,x5       #x11= 0x1
        xori x12,x7,0xAA    #x12= 0xAA
        srl x13,x5,x1      #X13=0x40000000
        andi x14,x8,0x1     #x14= 0x1
        or x15,x9,x3        #x15=0xFFFFFFFF
        add x16,x10,x10     #x16= 0x4
        xor x17,x11,x8      #x17= 0x1
        lw x18,0x4(x0)   #x18=0x80000000
        slt x19,x12,x4       #x19= 0
        srli x20,x13,0x1     #x20= 0x20000000
        and x21,x14,x6       #x21= 0
        sub x22,x5,x1        #x22= 0x7FFFFFFF
        addi x23,x10,0x1     #x23= 0x3
        or x24,x16,x9        #x24= 0xFFFFFFFB
        xor x25,x19,x11      #x25= 0x1
        andi x26,x20,0xFF    #x26= 0x200000FF
        add x27,x18,x3       #x27= 0x80000001
        srl x28,x20,x2       #x28= 0x10000000
        ori x29,x19,0xAF     #x29= 0xAF
        add x30,x20,x1       #x30= 0x20000001
        lw x31,0x8(x0)       #x31= 0x80000000
        jal x0,main
        add x0,x0,x0
        add x0,x0,x0
        add x0,x0,x0
```

执行顺序如何？

测试程序参考：

- 方案二： 有冒险的流水线测试(h.mem)

```
#baseAddr 0000
main:   addi x1,x0,0x1      #x1 = 0x1
        addi x2,x0,0x1      #x2 = 0x1
        addi x3,x0,0x1      #x3 = 0x1
        addi x4,x0,0x1      #x4 = 0x1
        lw x5,0x8(x0)       #x5 = 0x80000000
        add x6,x5,x1        #x6 = 0x80000001
        xor x7,x1,x2        #x7 = 0
        sub x8,x1,x7        #x8 = 0x1
        ori x9,x3,-1        #x9 = 0xFFFFFFFF
        and x10,x4,x3       #x10= 0x1
        sw x5,0x4(x0)   #mem(1)=0x8000000
        slt x11,x6,x5       #x11= 0x0
        xori x12,x7,0xAA    #x12= 0xAA
        beq x3,x8,loop1
        addi x0,x0,0x0
        add x0,x0,x0
loop1:  srl x13,x5,x1       #x13= 0x40000000
        andi x14,x8,0x1     #x14= 0x1
        or x15,x9,x3        #x15= 0xFFFFFFFF
        add x16,x10,x10     #x16= 0x2
        xor x17,x11,x8      #x17= 0x1
        lw x18,0x4(x0)      #x18= 0x80000000
        slt x19,x12,x4      #x19= 0
        srli x20,x13,0x1    #x20= 0x20000000
        and x21,x14,x10     #x21= 0x1
        bne x14,x12,loop2
        addi x0,x0,0x0
loop2:sub x22,x5,x1 #x22= 0x7FFFFFFF
        addi x23,x10,0x1    #x23= 0x2
        or x24,x16,x9       #x24= 0xFFFFFFFF
        xor x25,x19,x11     #x25= 0x0
        andi x26,x20,0xFF   #x26= 0x200000FF
        add x27,x18,x3      #x27= 0x80000001
        srl x28,x20,x2      #x28= 0x10000000
        ori x29,x19,0xAF    #x29= 0xAF
        add x30,x20,x1      #x30= 0x20000001
        lw x31,0x8(x0)      #x31= 0x80000000
        jal x0,main
        add x0,x0,x0
        add x0,x0,x0
        add x0,x0,x0
```

设计测试记录表格

- **ALU指令**测试结果记录
  - 自行设计记录表格


END