

Lab04-0

集成CPU核

-IP核设计CPU/IP2CPU

Ma De (马德)

made@zju.edu.cn

2020

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. 复习寄存器传输控制技术
2. 掌握CPU的核心组成：数据通路与控制
3. 设计数据通路的功能部件
4. 进一步了解计算机系统的基本结构
5. 熟练掌握IP核的使用方法

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. Sword2.0/Sword4.0开发板
3. Xilinx VIVADO2017.4及以上开发工具

□ 材料

无

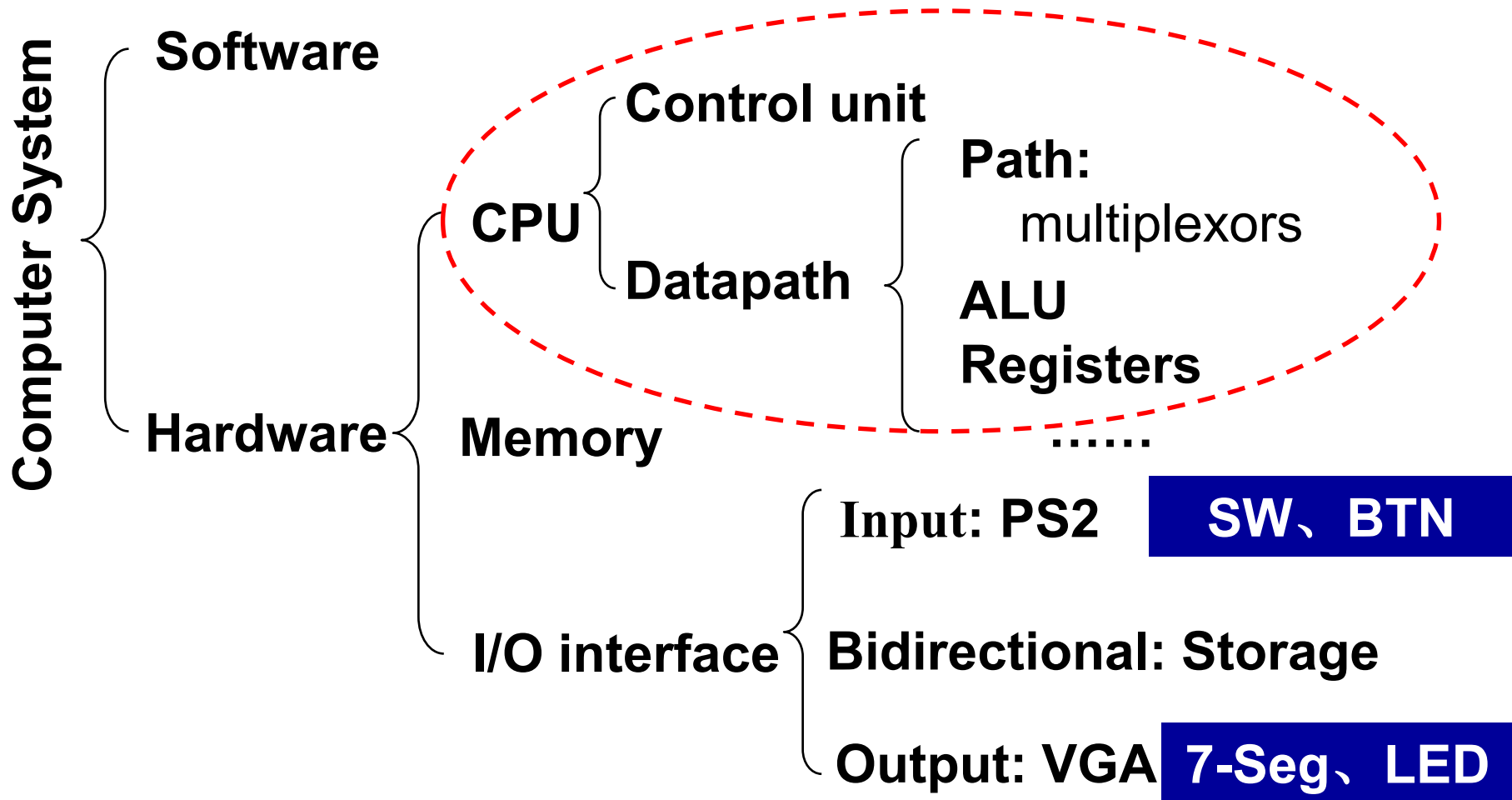
实验目标及任务

- **目标**：熟悉SOC系统的原理，掌握IP核集成设计CPU的方法
- **任务**：利用数据通路和控制器两个IP核集成设计CPU（采用原理图或RTL代码方式）

-
- 任务： 利用数据通路和控制器两个IP核集成设计CPU

Computer Organization

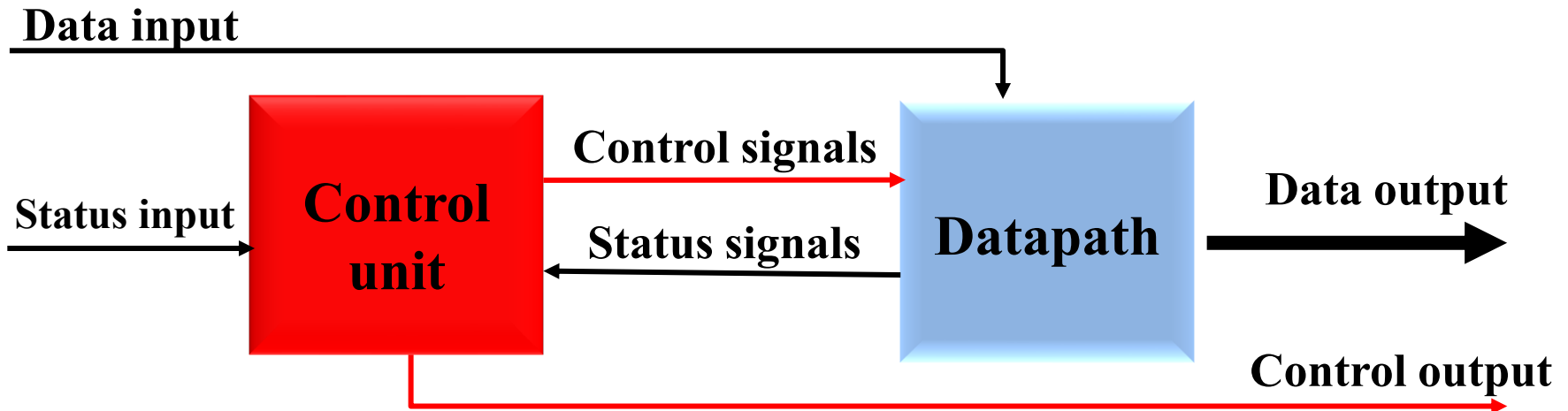
□ Decomposability of computer systems



Digital circuits vs CPU organization

□ Digital circuit

- General circuits that controls logical event with logical gates -
-Hardware



□ Computer organization

- Special circuits that processes logical action with instructions
-Software

CPU部件之1-数据通路: **Data_path**

□ Data_path

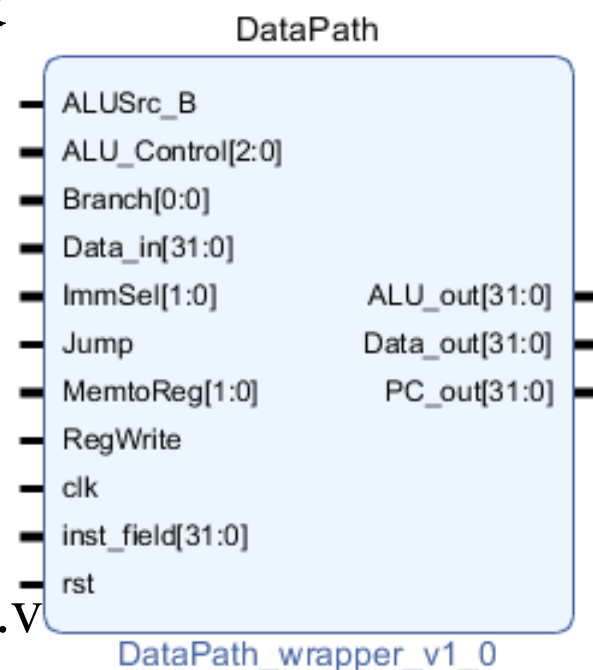
- CPU主要部件之一
- 寄存器传输控制对象: 通用数据通路

□ 基本功能

- 具有通用计算功能的算术逻辑部件
- 具有通用目的寄存器
- 具有通用计数所需的尽可能多的路径

□ 本实验用IP 软核- **Data_path**

- 核调用模块Data_path.v
- 核接口信号模块(空文档): Data_path.v



数据通路空模块- **Data_path.v**

```
module      Data_path( input clk,                //时钟
                        input rst,                //复位
                        input[31:0]inst_field,    //指令数据域[31:7]
                        input ALUSrc_B,
                        input [1:0]MemtoReg,
                        input [1:0] ImmSel,
                        input Jump,
                        input Branch,
                        input RegWrite,
                        input[31:0]Data_in,
                        input[2:0]ALU_Control,

                        output[31:0]ALU_out,
                        output[31:0]Data_out,
                        output[31:0]PC_out
                        );

endmodule
```

CPU部件之2-控制器： SCPU_ctrl

□ SCPU_ctrl

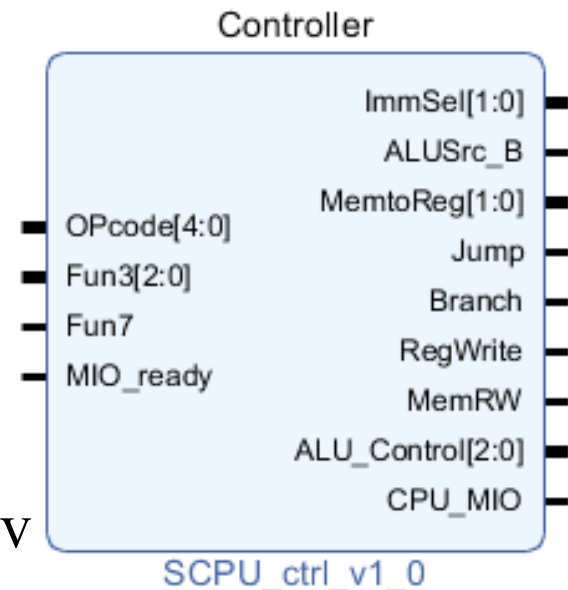
- CPU主要部件之一
- 寄存器传输控制技术中的运算和通路控制器：

□ 基本功能

- 指令译码
- 产生操作控制信号：ALU运算控制
- 产生指令所需的路径选择

□ 本实验用IP 软核- SCPU_ctrl

- 核调用模块SCPU_ctrl.v
- 核接口信号模块(空文档)： SCPU_ctrl.v



控制器接口文档- **SCPU_ctrl.v**

```
module      SCPU_ctrl( input[4:0]OPcode,    //Opcode—inst[6:2]
                        input[2:0]Fun3,      //Function-inst[14:12]
                        input    Fun7,      //Function-inst[30]
                        input MIO_ready,    //CPU Wait
                        output reg [1:0] ImmSel//立即数选择
                        output reg ALUSrc_B,
                        output reg[1:0]MemtoReg,
                        output reg Jump,
                        output reg Branch,
                        output reg RegWrite,
                        output reg MemRW,
                        output reg [2:0]ALU_Control,
                        output reg CPU_MIO
                        );

endmodule
```

方法一

逻辑原理图集成设计CPU

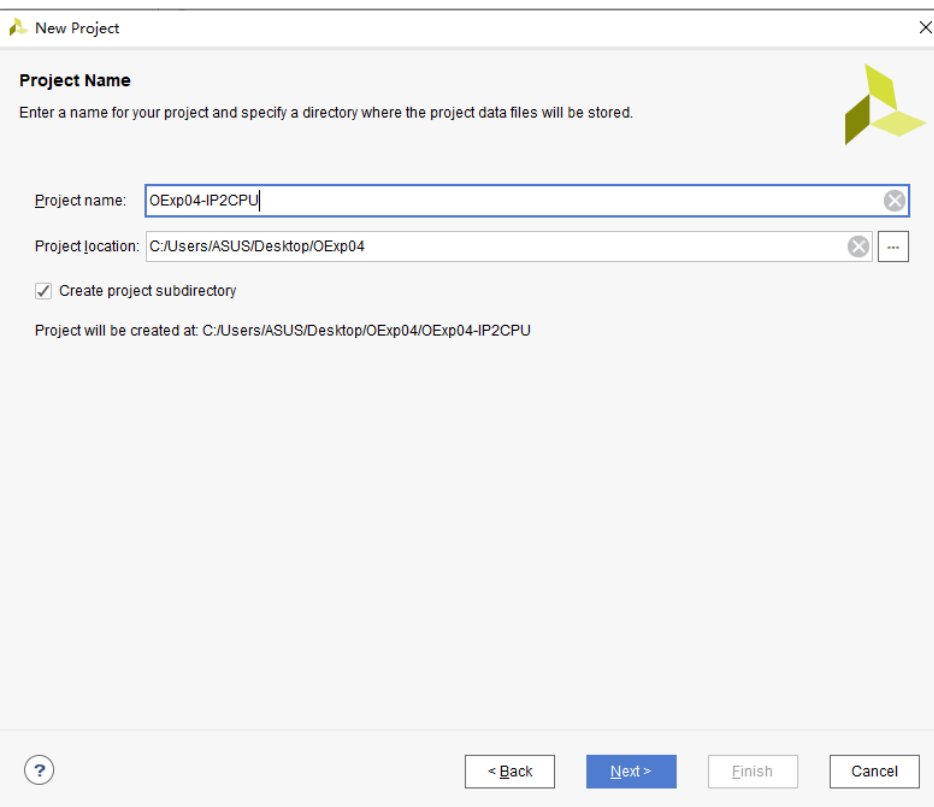
设计工程：OExp04-IP2CPU

◎ 分解CPU为二个IP核

④ 用二个IP核构建CPU

④ 顶层工程名称沿用OExp04-IP2CPU

◎ BD模块名：IP2CPU.bd



New Project

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

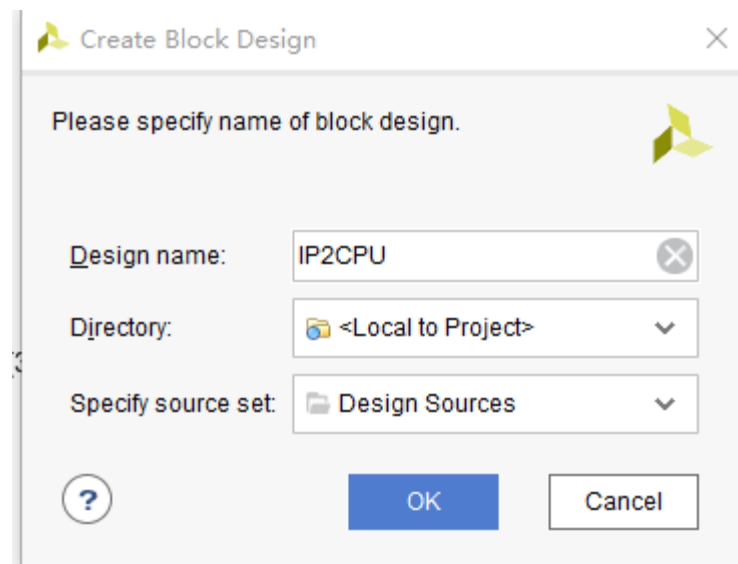
Project name: OExp04-IP2CPU

Project location: C:/Users/ASUS/Desktop/OExp04

☒ Create project subdirectory

Project will be created at: C:/Users/ASUS/Desktop/OExp04/OExp04-IP2CPU

< Back Next > Finish Cancel



Create Block Design

Please specify name of block design.

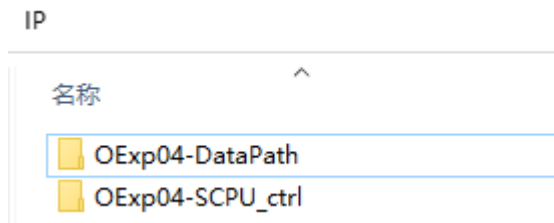
Design name: IP2CPU

Directory: <Local to Project>

Specify source set: Design Sources

? OK Cancel

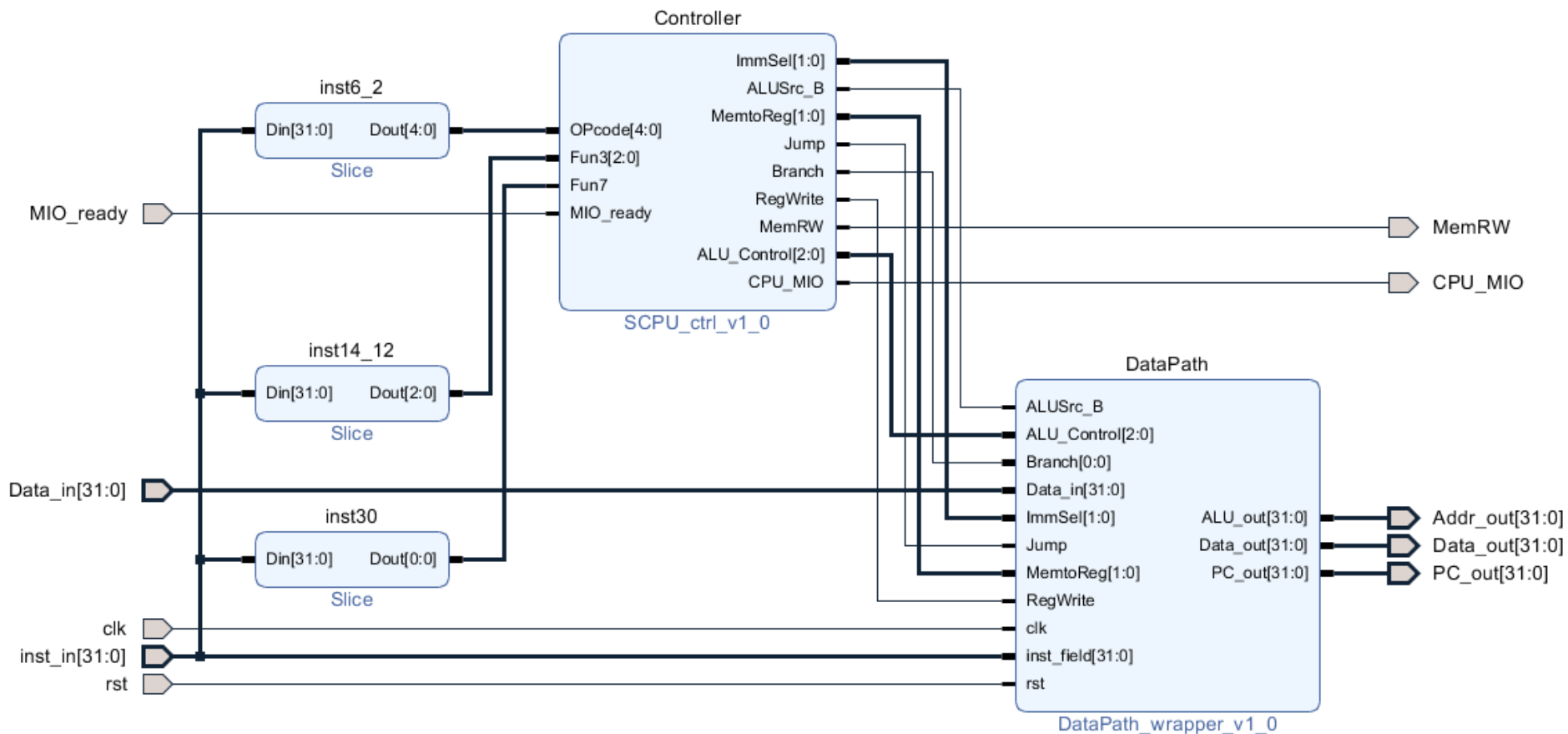
添加二个核的封装文件到当前工程IP库目录：



BD工程中增加SCPU_ctrl.xci、 Data_path.xci

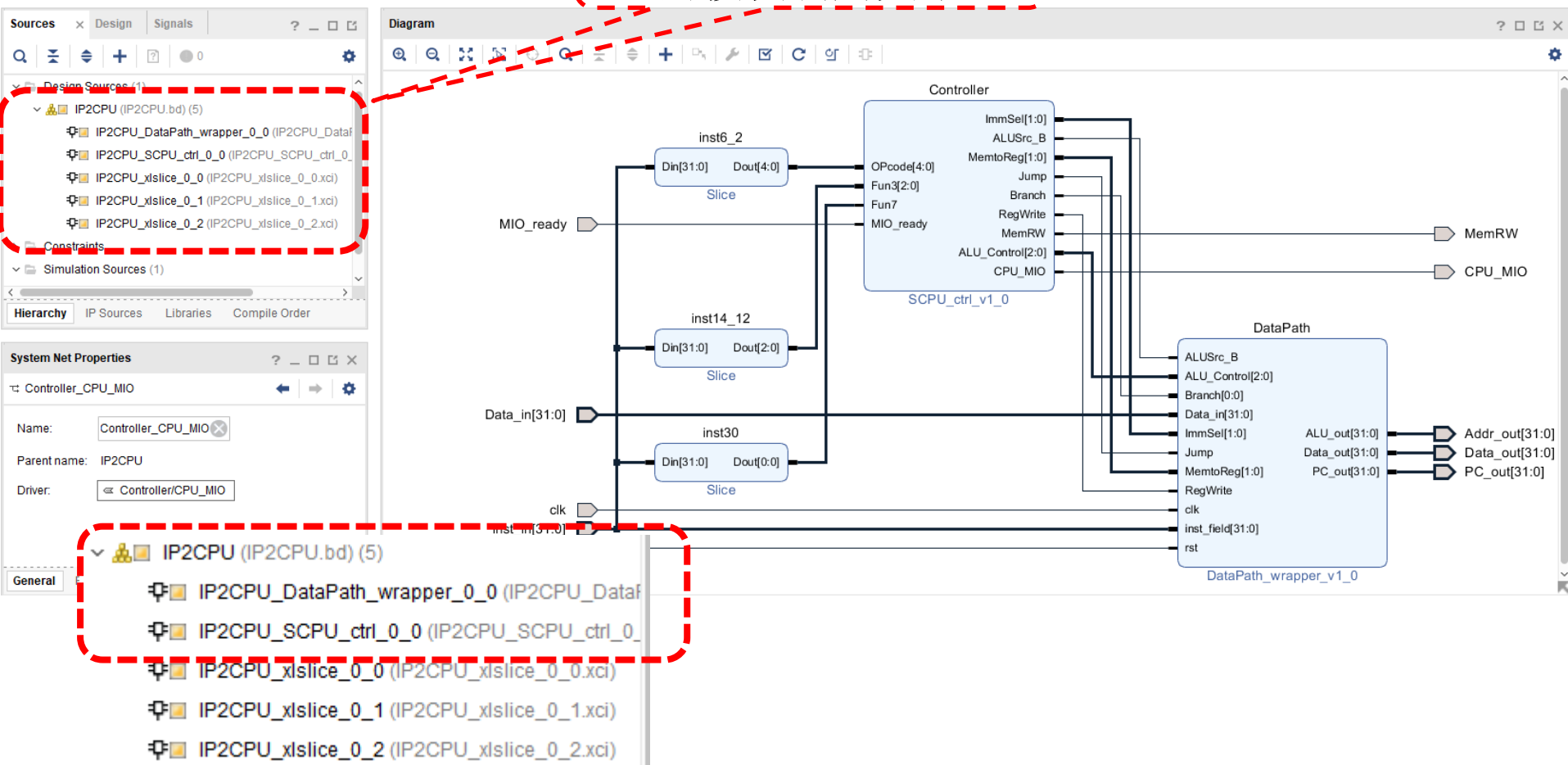
🔗 IP2CPU_DataPath_wrapper_0_0 (IP2CPU_DataPath_wrapper_0_0.xci)
🔗 IP2CPU_SCPU_ctrl_0_0 (IP2CPU_SCPU_ctrl_0_0.xci)

用逻辑原理图输入CPU设计

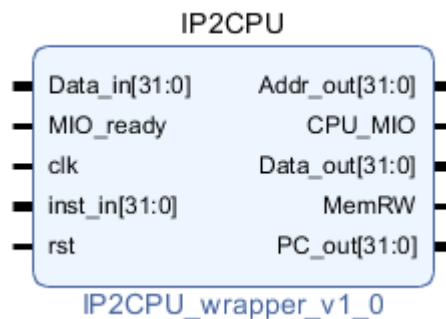
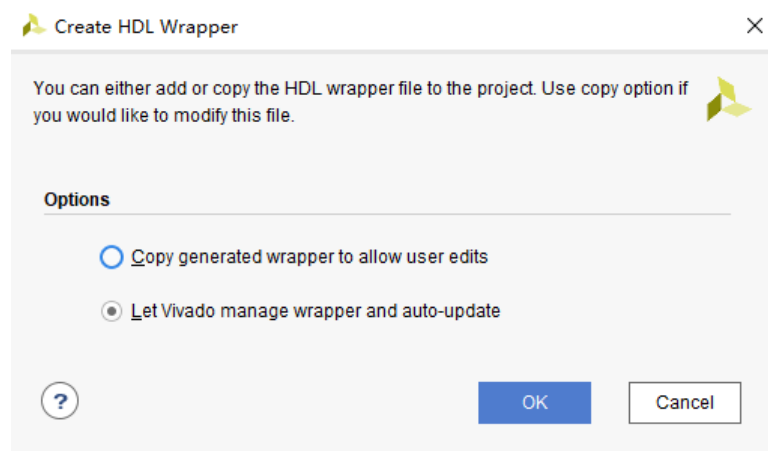
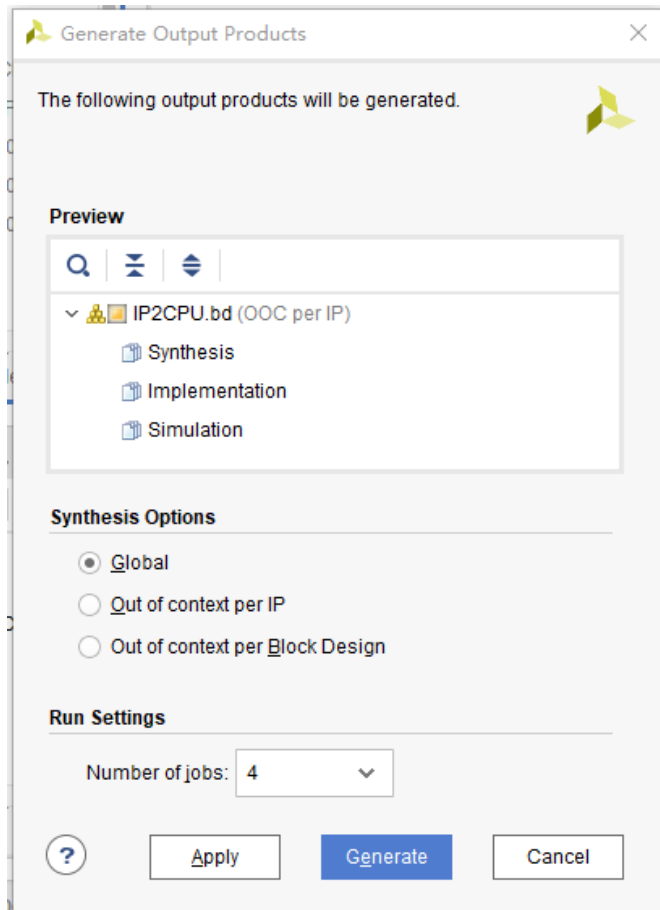


CPU集成设计后的层次关系

Exp04完成CPU设计后的
模块调用关系



CPU核的生成

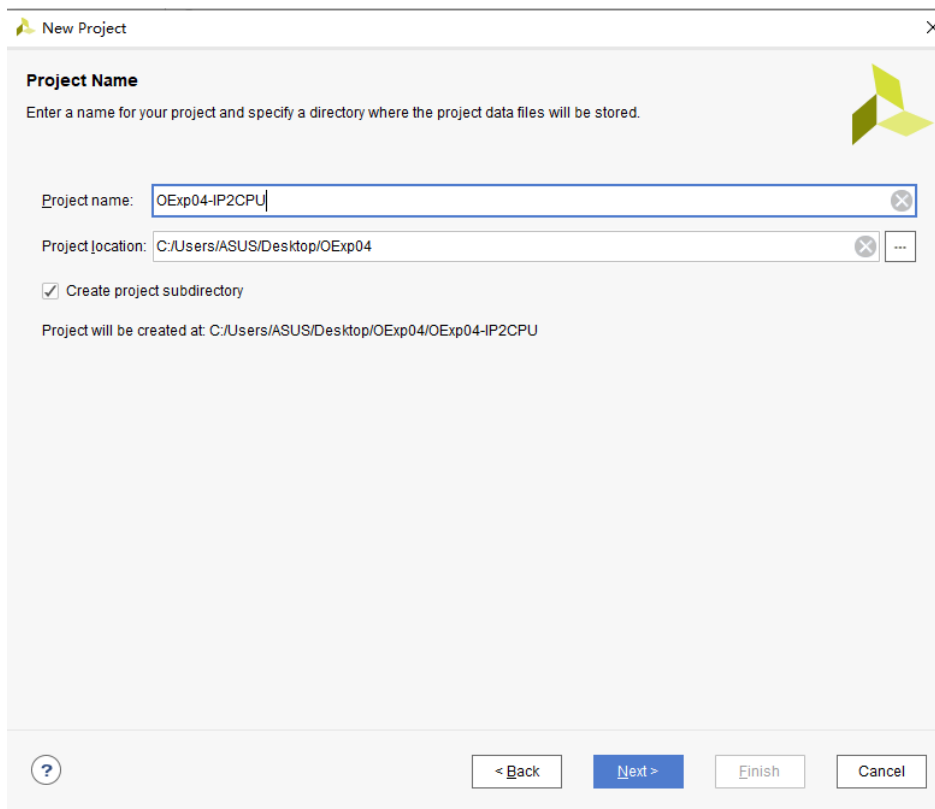


方法二

RTL集成设计实现CPU

设计工程：OExp04-IP2CPU

- ◎ 工程的创建和IP核添加与原理图方式相同
- ◎ 创建顶层文件SCPU.v



New Project

Project Name
Enter a name for your project and specify a directory where the project data files will be stored.

Project name: OExp04-IP2CPU

Project location: C:/Users/ASUS/Desktop/OExp04

☒ Create project subdirectory

Project will be created at: C:/Users/ASUS/Desktop/OExp04/OExp04-IP2CPU

? < Back Next > Finish Cancel

用RTL代码描述CPU设计 SCPU.V

- 利用verilog描述，在SCPU.v顶层进行模块调用和连接

```
module SCPU(  
    input wire clk,  
    input wire rst,  
    input wire MIO_ready,  
    input wire[31:0] inst_in,  
    input wire[31:0] Data_in,  
  
    output wire CPU_MIO,  
    output wire MemRW,  
    output wire[31:0] PC_out,  
    output wire[31:0] Data_out,  
    output wire[31:0] Addr_out  
);
```

```
wire [1:0] ImmSel;  
wire      ALUSrc_B;  
wire [1:0] MemtoReg;  
wire      Jump;  
wire      Branch;  
wire      RegWrite;  
wire [2:0] ALU_Control;
```

Wire变量用于模块连接，
与原理图的导线类似

SCPU_ctrl U1(
 .OPcode (inst_in[6:2]),
 .Fun3 (inst_in[14:12]),
 .Fun7 (inst_in[30]),
 .MIO_ready (MIO_ready),
 .ImmSel (ImmSel),
 .ALUSrc_B (ALUSrc_B),
 .MemtoReg (MemtoReg),
 .Jump (Jump),
 .Branch (Branch),
 .RegWrite (RegWrite),
 .MemRW (MemRW),
 .ALU_Control (ALU_Control),
 .CPU_MIO (CPU_MIO)

IP模块例化，与原理图的
BD模块类似

CPU集成设计后的层次关系

The screenshot displays the project hierarchy and source code for a CPU design. The 'Sources' window on the left shows the design structure, with 'SCPU (SCPU.v) (2)' highlighted. The 'Source File Properties' window below it shows the 'General' tab for 'SCPU.v'. The 'Project Summary' window on the right shows the file path and the source code for 'SCPU.v'.

Sources

- Design Sources (1)
 - SCPU (SCPU.v) (2)
 - U1: SCPU_ctrl (SCPU.v)
 - U2: DataPath (DataPath.v)

Source File Properties

SCPU.v

Project Summary

SCPU.v

C:/Users/ASUS/Desktop/OExp_RISCv/OExp04/basic/IP/CPU/SCPU (

```
37     wire    ALUSrc_B;  
38     wire [1:0] MemtoReg;  
39     wire    Jump;  
40     wire    Branch;  
41     wire    RegWrite;  
42     wire [2:0] ALU_Control1;  
43  
44     SCPU_ctrl U1(  
45         .OPcode      (inst_in[6
```

Question: lab02—SOC环境中RAM的生成

□ RAM的生成

- 若直接使用提供的IP库里的RAM_B则读写正常
- 若采用lab01的方法生成的RAM则可能出现错误
- 如图所示，默认的输出端口会勾选输出寄存器，导致数据会往后延迟一个时钟周期，所以应该不勾选

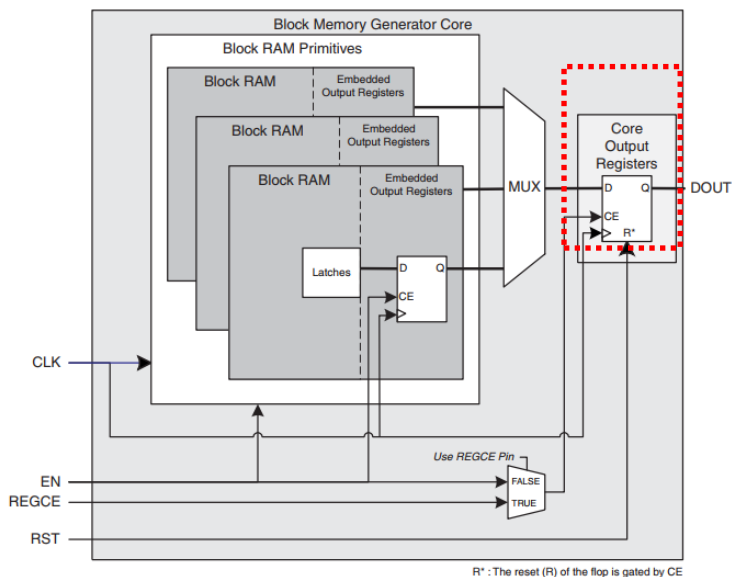
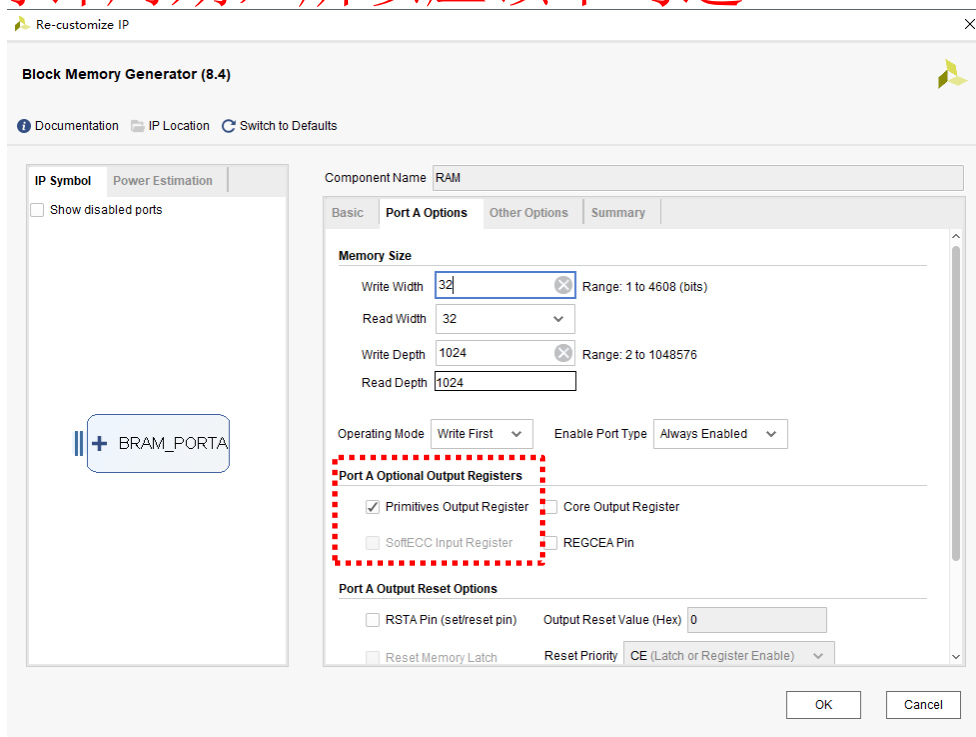


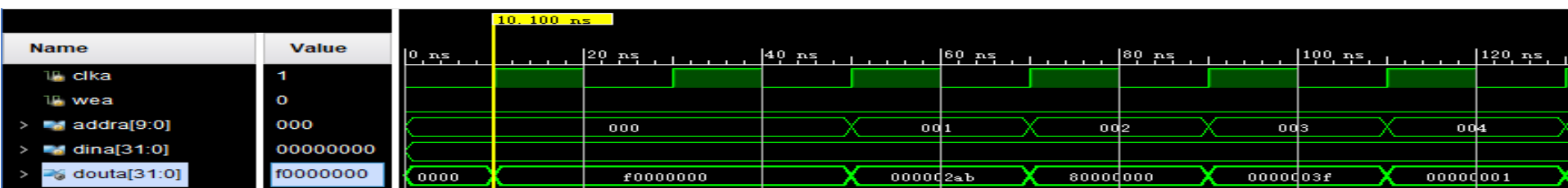
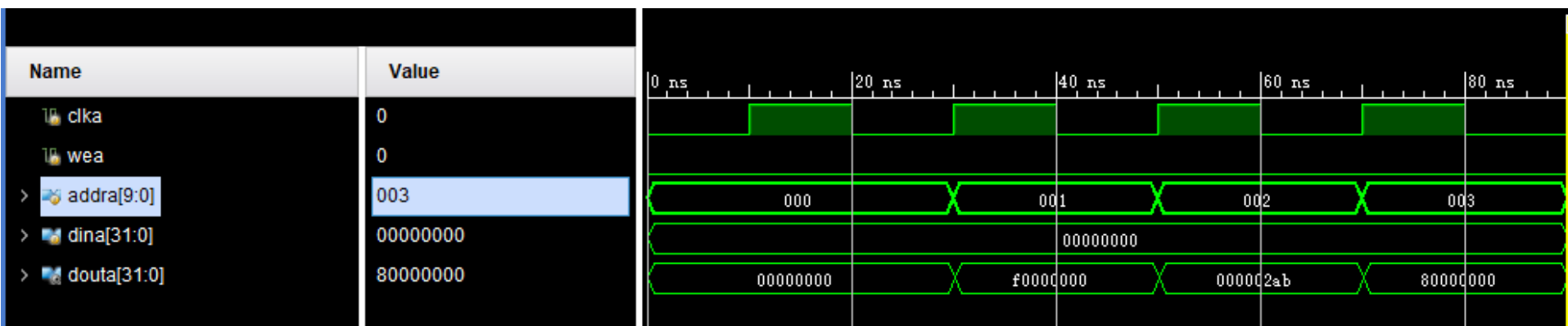
Figure D-2: Block Memory Generated with Register Port [A][B] Output of Memory Primitives and Register Port [A][B] Output of Memory Core Enabled



Question: lab02—SOC环境中RAM的生成

- 波形图1为勾选输出寄存器，读出数据延迟一拍
- 波形图2为不勾选输出寄存器，正常读出数据

```
memory_initialization_radix=16;  
memory_initialization_vector=  
f0000000, 000002AB, 80000000, 0000003F, 00000001, FFF70000, 0000FFFF, 80000000, 00000000, 11111111,  
22222222, 33333333, 44444444, 55555555, 66666666, 77777777, 88888888, 99999999, aaaaaaaa, bbbbbbbb,  
cccccccc, dddddddd, eeeeeeee, ffffffff, 557EF7E0, D7BDFBD9, D7DBFDB9, DFCFFCFB, DFCFBFFF, F7F3DFFF,  
FFFFDF3D, FFFF9DB9, FFFFBCFB, DFCFFCFB, DFCFBFFF, D7DB9FFF, D7DBFDB9, D7BDFBD9, FFFF07E0, 007E0FFF,  
03bdf020, 03def820, 08002300;
```



 **END**