计算机设计与实践

单周期CPU设计(IF、ID)

2022·夏

HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ

实验目的

- 通过模块化设计方式，加深对CPU结构和工作原理的理解
- 掌握根据**数据通路表**和**控制信号取值表**来实现*取指*、*译码*单元的方法
- 熟练掌握使用Verilog HDL实现CPU的功能部件

实验内容

- 使用Verilog HDL实现单周期CPU的**取指**单元：
  - 根据数据通路表和控制信号取值表，确定取指单元的**接口**
  - 实现所需的功能部件（如PC、NPC、IROM，etc.）

所属单元	取指单元			
部件	PC	NPC		IROM
输入信号	din	PC	imm	adr

  - 根据数据通路表，连接各个部件，形成取指单元

实验内容

- 使用Verilog HDL实现单周期CPU的**译码**单元：
  - 根据数据通路表和控制信号取值表，确定译码单元的**接口**
  - 实现所需的功能部件（如RF、SEXT，etc.）

译码单元				
RF				SEXT
rR1	rR2	wR	wD	din

  - 根据数据通路表，连接各个部件，形成译码单元

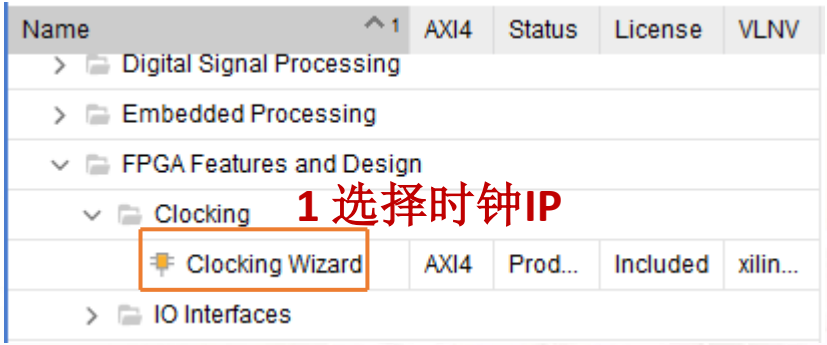
实验原理 – 模块化设计

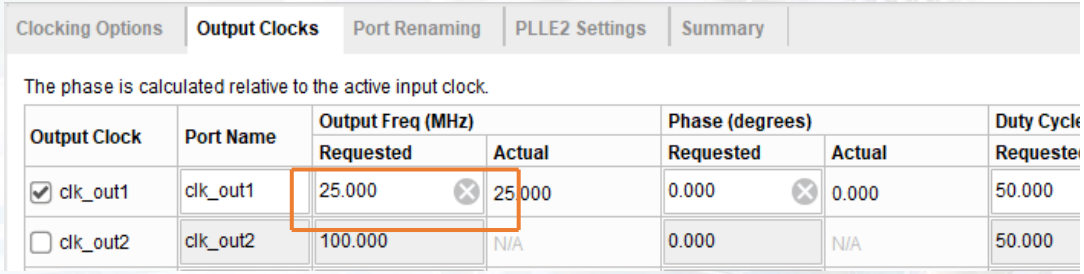
```
module miniRV(rst_i, clk_i, debug信号);
    input rst_i;            // 板上的Reset信号，低电平复位
    input clk_i;            // 板上的100MHz时钟信号
```

模 块	文 件	备 注
顶层模块	**miniRV.v**	**实例化、连接各部件**
时钟	**cpuclk.v**	**系统时钟 (25MHz)**
存储器模块	**prgrom.v**	**指令存储器 (64KB)**
	dmem.v	**数据存储器 (64KB)**
取指模块	*ifetch.v*	
译码模块 (含寄存器组)	*idecode.v*	
执行模块	*execute.v*	
控制器模块	*control.v*	

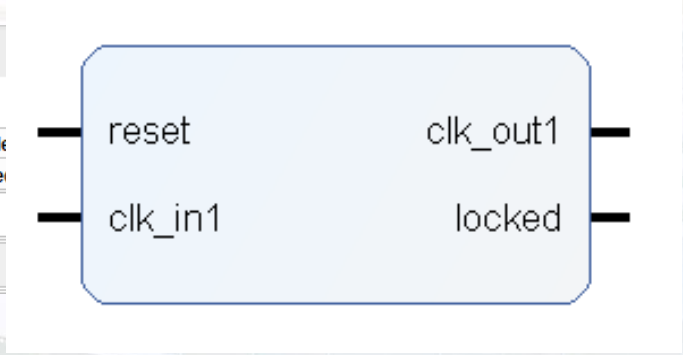
示例

时钟模块实现 —— 使用IP核

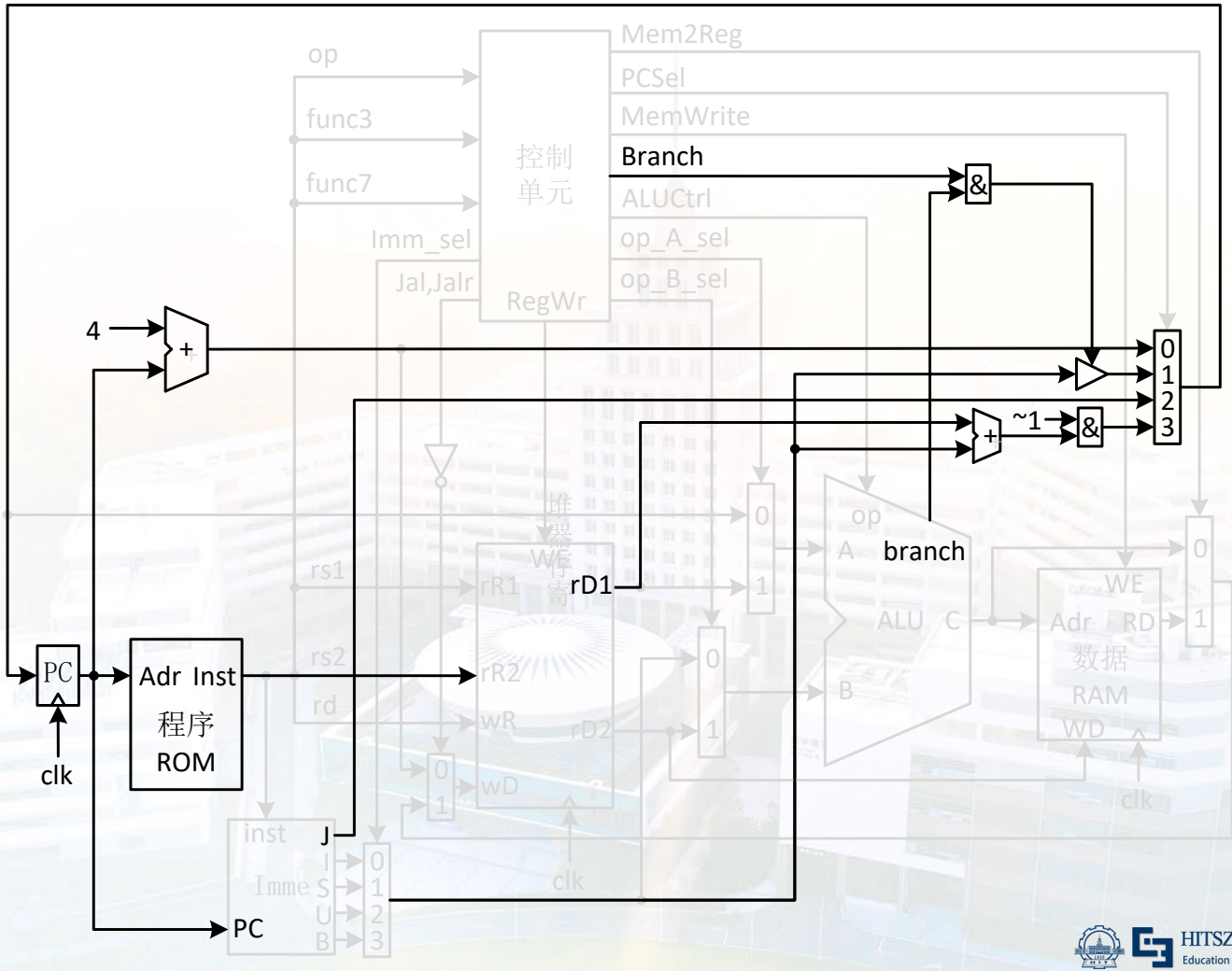




3 选择输出时钟频率



取指单元实现


op
func3
func7
Imm_sel
Jal,Jalr
控制单元
RegWr
Mem2Reg
PCSel
MemWrite
Branch
ALUCtrl
op_A_sel
op_B_sel
4
+
&
~1
0
1
2
3
PC
clk
Adr Inst
程序
ROM
rs1
rs2
rd
rR1
rR2
wR
wD
rD1
rD2
堆栈寄存器
op
A
B
ALU
C
branch
WE
Adr
RD
数据
RAM
WD
inst
J
Imme
I
S
U
B
PC

取指单元实现 – 关键点

- 使用Distributed Memory IP核建立IROM (类似时钟IP核，详见指导书)
- 利用PC，从IROM中取出指令
- 对PC值进行+4
- 完成几种跳转指令的PC修改功能
  - 要点：根据其他单元的关键信号，用ALU计算结果or立即数修改PC
- 最终修改PC值
  - 要点：使用多路选择器

取指单元实现 – 基本思路

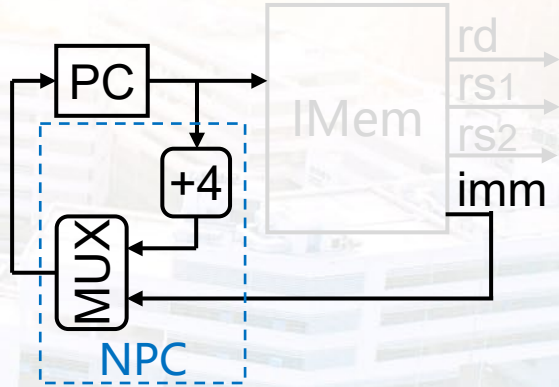
- 首先实现取指单元所需的功能部件
- 根据数据通路表的连接关系，将各部件连接起来

所属单元	取指单元			
部件	PC	NPC		IROM
输入信号	din	PC	imm	adr
add	NPC.npc	PC.pc		PC.pc
sub	NPC.npc	PC.pc		PC.pc
ori	NPC.npc	PC.pc		PC.pc
lw	NPC.npc	PC.pc		PC.pc
sw	NPC.npc	PC.pc		PC.pc
beq	NPC.npc	PC.pc	SEXT.ext	PC.pc
jal	NPC.npc	PC.pc	SEXT.ext	PC.pc

- 封装成模块，得到取指单元

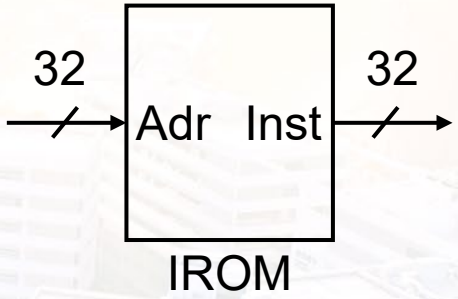
构建功能部件 – PC和NPC

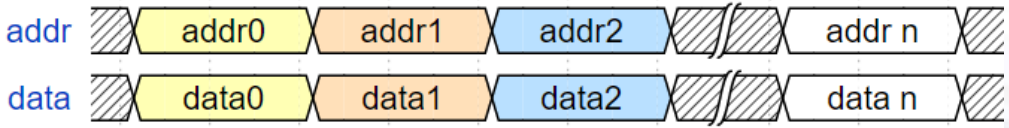
- PC：32bit寄存器，存储着当前指令的地址
  - PC的BIT1和BIT0恒为0，故也可使用30bit寄存器
  - PC的初始值是CPU复位后执行的首条指令的地址
  - 要点：用NPC.npc更新PC
- NPC：
  - 输入：PC、imm
  - 2种操作：PC+4、PC+sext(imm)
  - 要点：根据操作码、功能码判断所需操作



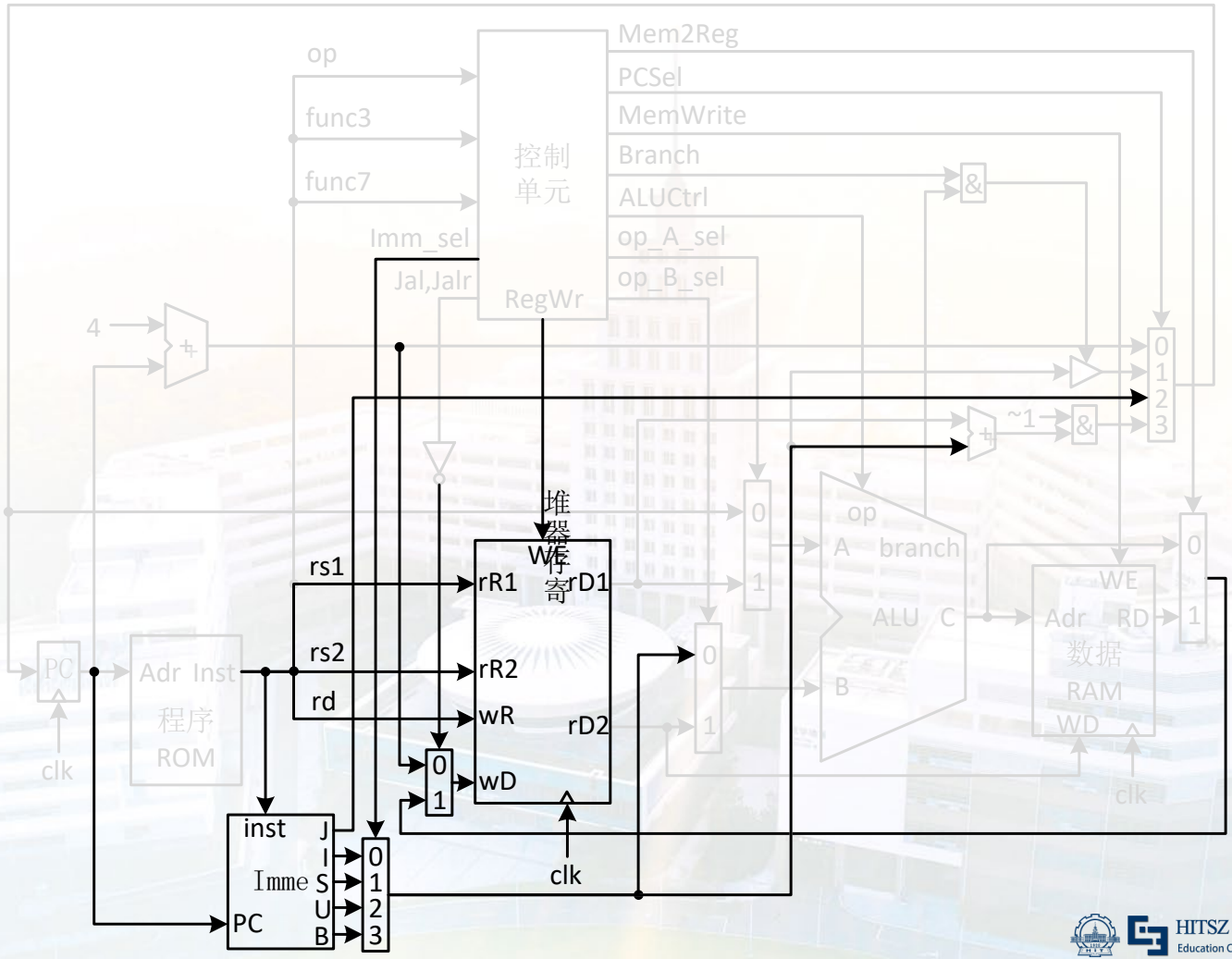
构建功能部件 – IROM

- IROM是一个单端口的只读存储器
  - 输入地址：PC
  - 输出数据：指令
- IROM的读取操作是组合逻辑
  - 读时序：



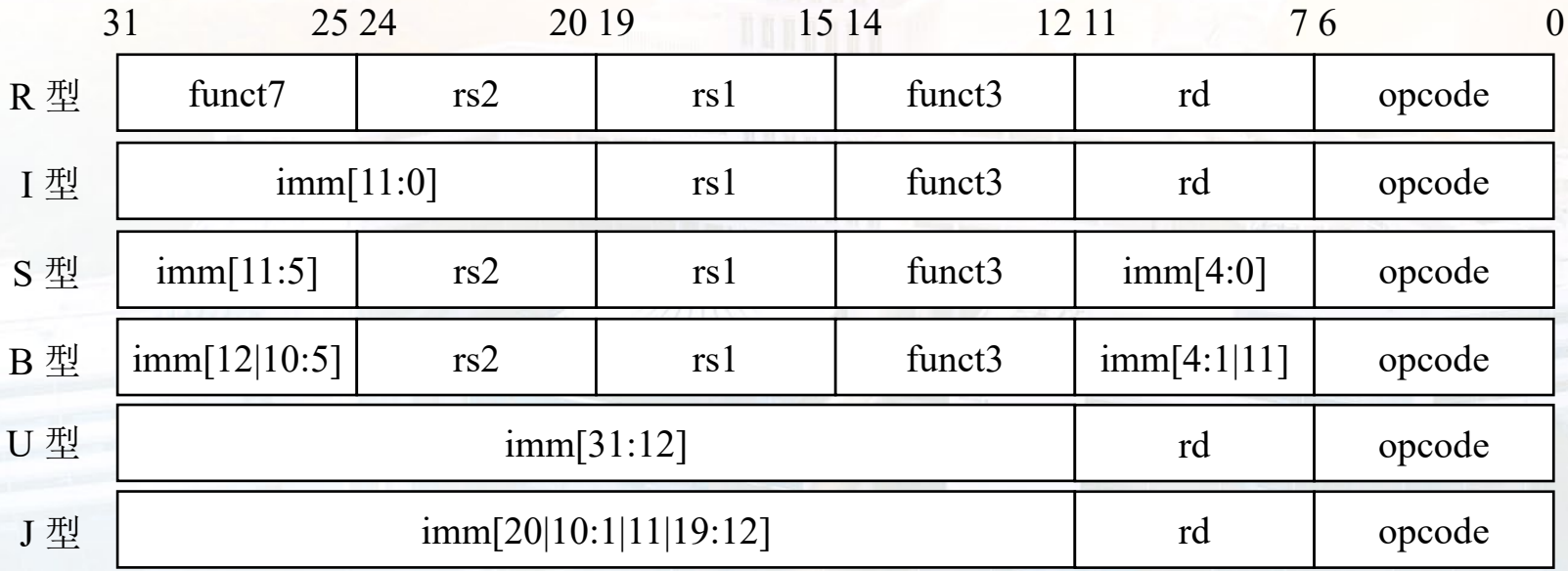


译码单元实现


op
func3
func7
Imm_sel
Jal,Jalr
控制单元
RegWr
Mem2Reg
PCSel
MemWrite
Branch
ALUCtrl
op_A_sel
op_B_sel
4
&
~1
0
1
2
3
堆栈寄存器
WE
rs1
rs2
rd
rR1
rR2
wR
wD
rD1
rD2
clk
PC
Adr Inst
程序ROM
op
A branch
ALU C
B
WE
Adr RD
数据RAM
WD
inst
Imme
J
I
S
U
B
PC

译码单元实现 – 关键点

- 分解指令中的各个字段 —— 按照指令格式，分别取出各个字段
  - ✓ Tips：用**assign**语句、位选择符**[]**、位拼接符**{}**实现



译码单元实现 – 关键点

- 分解指令中的各个字段
- 建立寄存器文件RF
- 对寄存器文件进行读写操作
  - 要点：根据分解指令得到的寄存器号和来自其他单元的关键信号，读写相应寄存器
- 对立即数进行符号扩展
  - 要点：lui、jal指令的立即数是20位，其余是12位

    不同类型的指令，其立即数扩展操作不同

译码单元实现 – 基本思路

- 根据指令格式，分解指令中的各个字段
- 实现译码单元所需的各个部件
- 根据数据通路表的连接关系，将各部件连接起来

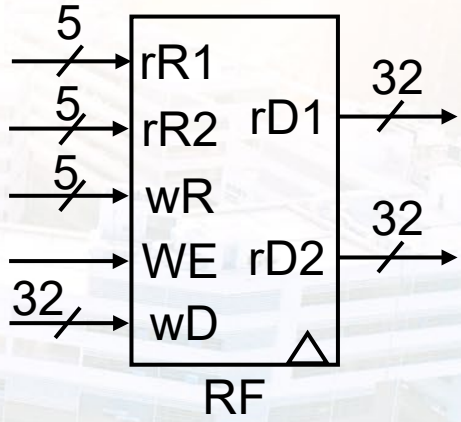
译码单元				
RF				SEXT
rR1	rR2	wR	wD	din
IROM.inst[19:15]	IROM.inst[24:20]	IROM.inst[11:7]	ALU.C	
IROM.inst[19:15]	IROM.inst[24:20]	IROM.inst[11:7]	ALU.C	
IROM.inst[19:15]		IROM.inst[11:7]	ALU.C	IROM.inst[31:20]
IROM.inst[19:15]		IROM.inst[11:7]	DRAM.rd	IROM.inst[31:20]
IROM.inst[19:15]	IROM.inst[24:20]			IROM.inst[31:25|11:7]
IROM.inst[19:15]	IROM.inst[24:20]			IROM.inst[31|7|30:25|11:8]
		IROM.inst[11:7]	NPC.pc4	IROM.inst[31|19:12|20|30:21]

- 封装成模块，得到译码单元

构建功能部件 – RF

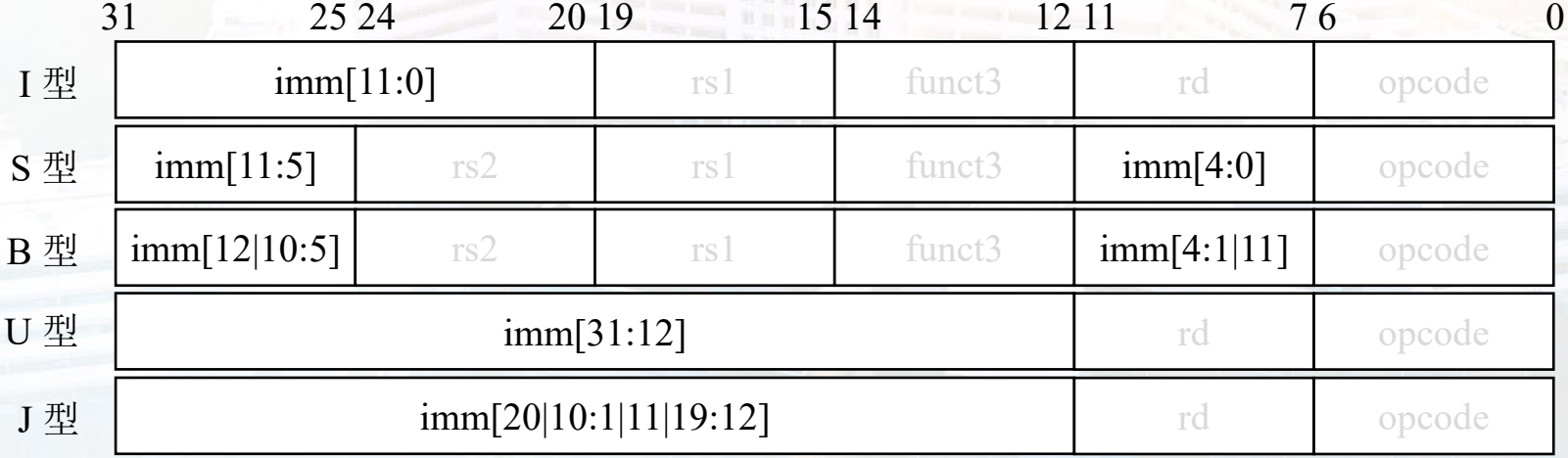
- RF包含32个32bit寄存器
  - 1条指令最多有2个源寄存器(rs1、rs2)、1个目标寄存器(rd)
  - 因此，RF需要3个“地址”端口、3个“数据”端口

- RF的读写逻辑：
  - WE=0: rD1<-REG[rR1]， rD2<-REG[rR2]
  - WE=1: REG[wR] <- wD



构建功能部件 – SEXT

- SEXT是符号扩展部件：
  - 输入：指令中的立即数 (12bit或20bit)
  - 输出：符号扩展之后的立即数 (32bit)
- 要点：根据指令中立即数的格式进行扩展



实验步骤

① 根据数据通路表，确定取指、译码单元**包含哪些部件**

② 根据数据通路表，确定各功能部件的**接口**和需要实现的**功能**

③ 使用Verilog HDL实现各功能部件

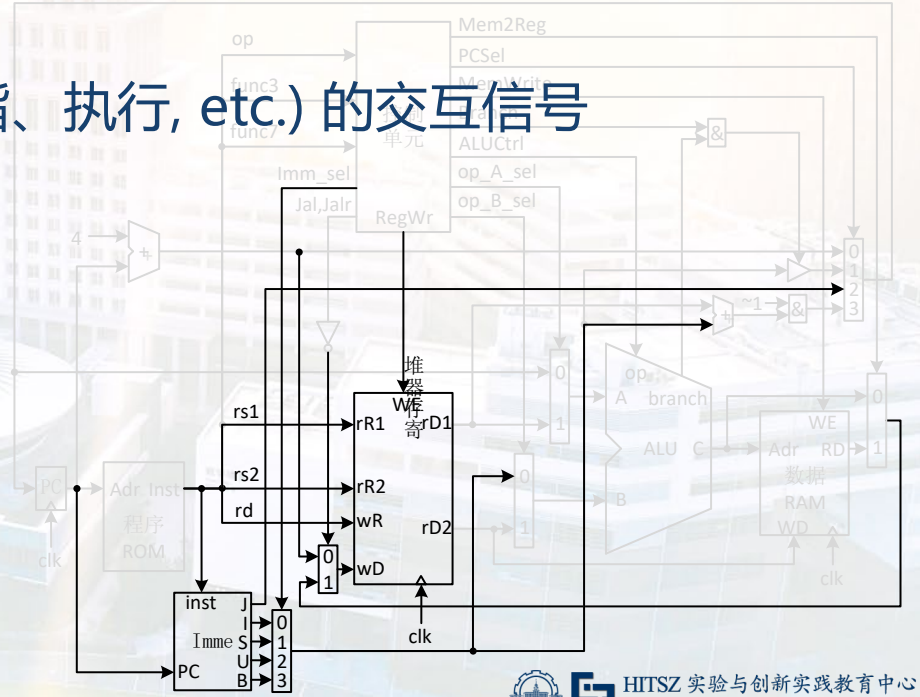
④ 根据数据通路表，将各功能**部件连接**起来

⑤ **封装**成模块，得到取指单元和译码单元

课堂检查

- 将自己设计的译码单元画出来
- 译码单元图需包括：
  - 译码单元与其他单元 (取指、执行, etc.) 的交互信号
  - 内部子模块
  - 内部子模块之间的连接

参考示例：



所画的图要和数据通路表对应！

开始实验