计算机设计与实践

单周期CPU设计(IF、ID)

2025 · 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：
  - 从数据通路表和控制信号取值表提取出IF、ID单元的各部件

单元	取指单元				
部件	PC	NPC			IROM
输入	din	PC	offset	br	adr

译码单元				
RF				SEXT
rR1	rR2	wR	wD	din

  - 实现所需的功能部件（如PC、NPC、IROM；RF、SEXT，etc.）
  - 根据数据通路表，连接各个部件，得到IF、ID单元

实验原理 – 模块化设计

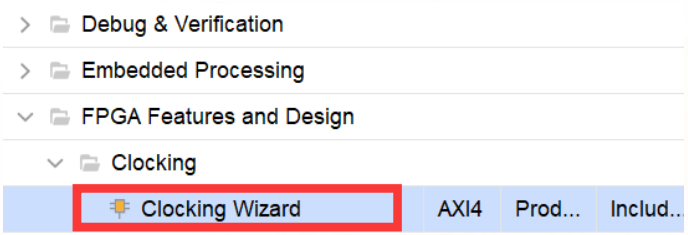
```
module miniRV_SoC(fpga_rst, fpga_clk, 外设接口信号, debug信号);
    input fpga_rst;          // 板上的Reset信号，高电平复位
    input fpga_clk;          // 板上的100MHz时钟信号
```

模 块	文 件	备 注
顶层模块	**miniRV_SoC.v**	**实例化、连接各部件**
时钟	**cpuclk.xci**	**PLL时钟IP核 (25MHz)**
CPU	**myCPU.v**	TODO：实例化、连接各单元/部件
存储器模块	**IROM.xci**	**指令存储器IP核 (64KB)**
	DRAM.xci	**数据存储器IP核 (64KB)**
总线桥	**Bridge.v**	
IF模块	*ifetch.v*	*可以按功能单元划分子模块*
ID模块	*idecode.v*	*也可按功能部件划分子模块*
EX模块	*execute.v*	
控制器模块	*control.v*	

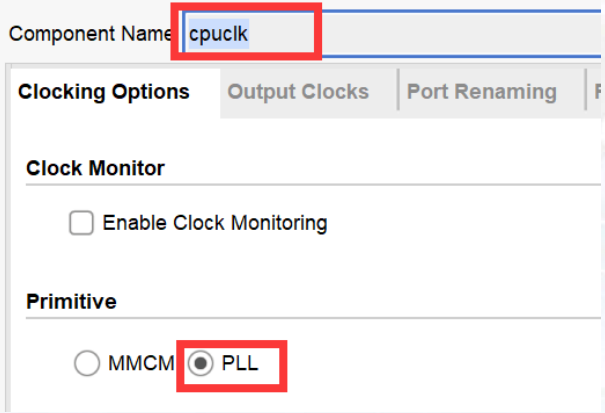
示例

时钟模块实现 —— 使用IP核

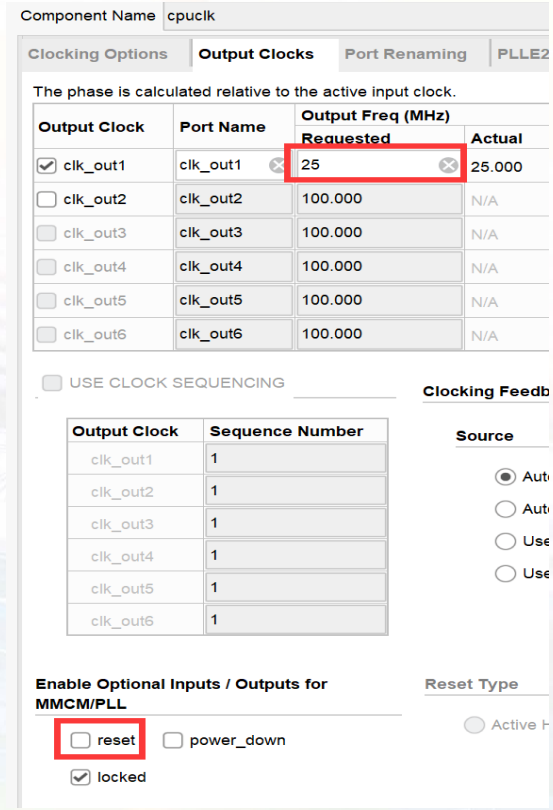
a. 选择时钟IP核

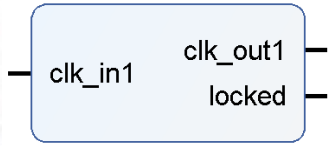


b. 修改模块名、选择时钟类型



c. 设置频率、去掉reset

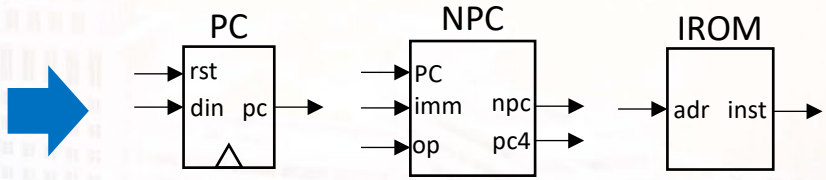




IF单元实现 – 基本思路

- 综合考虑2个设计表，从中提取各部件，明确接口

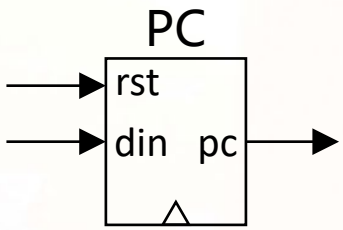
所属单元	取指单元			
部件	PC	NPC		IROM
输入信号	din	PC	imm	adr
add	NPC.npc	PC.pc		PC.pc
sub	NPC.npc	PC.pc		PC.pc
ori	NPC.npc	PC.pc		PC.pc
lw	NPC.npc	PC.pc		PC.pc
sw	NPC.npc	PC.pc		PC.pc
beq	NPC.npc	PC.pc	SEXT.ext	PC.pc
jal	NPC.npc	PC.pc	SEXT.ext	PC.pc



- 实现取指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
- 封装成模块，得到取指单元

IF单元实现 – PC

- PC是一个32bit寄存器，存储着当前指令的地址
  - 指令均为32bit定长，即每条指令4Byte
  - 因此PC的BIT1和BIT0恒为0，故也可使用30bit寄存器
- PC的更新：
  - **指令正常执行**：pc ← din
  - **CPU复位时**：pc ← 0x0H



部件	信号名	属性	释义
PC	rst	输入	复位信号
	clk	输入	时钟信号
	din	输入	下一条指令地址
	pc	输出	当前指令地址

IF单元实现 – NPC

- NPC具有两个用途：
  - ① pc4输出PC+4的值（jal的写回）

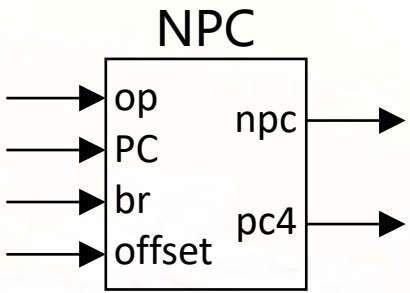
    pc4 ← PC+4
  - ② npc输出下一条指令地址

    顺序执行：npc ← PC+4

    条件分支：npc ← br ? PC+offset

    : PC+4

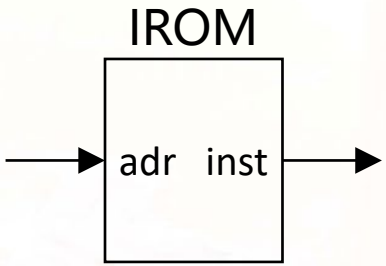
    无条件跳转：npc ← PC+offset



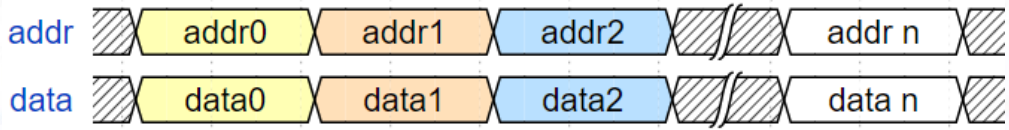
部件	信号名	属性	释义
NPC	op	输入	控制npc的产生
	PC	输入	当前指令地址
	br	输入	条件分支是否跳转
	offset	输入	跳转的偏移量
	pc4	输出	PC+4的值
	npc	输出	下一条指令地址

IF单元实现 – 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支持异步读（读操作是组合逻辑）
  - 读时序：



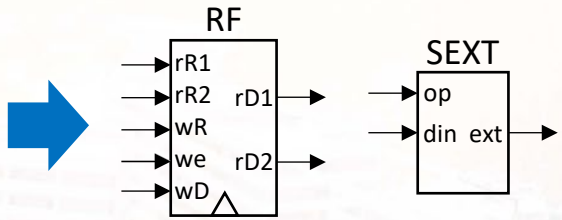
IROM	adr	输入	指令地址
	inst	输出	指令



ID单元实现 – 基本思路

- 分解IF单元取出的指令
- 综合考虑2个设计表，从中提取各部件，明确接口

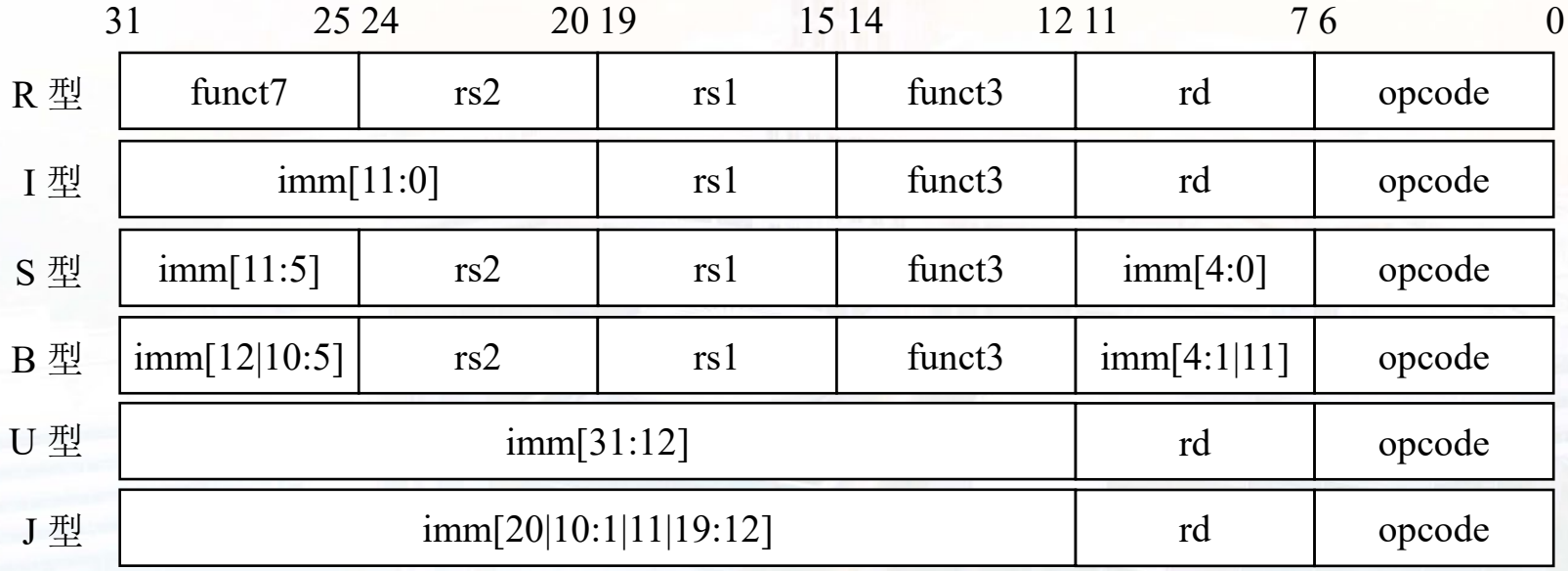
单元	译码单元				
部件	RF				SEXT
输入	rR1	rR2	wR	wD	din
add	IROM.inst[19:15]	IROM.inst[24:20]	IROM.inst[11:7]	ALU.C	-
ori	IROM.inst[19:15]	-	IROM.inst[11:7]	ALU.C	IROM.inst[31:20]
sw	IROM.inst[19:15]	IROM.inst[24:20]	-	-	IROM.inst[31:25|11:7]
beq	IROM.inst[19:15]	IROM.inst[24:20]	-	-	IROM.inst[31|7|30:25|11:8]
lui	-	-	IROM.inst[11:7]	SEXT.ext	IROM.inst[31:12]
jal	-	-	IROM.inst[11:7]	NPC.pc4	IROM.inst[31|19:12|20|30:21]



- 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
- 封装成模块，得到译码单元

ID单元实现 – 分解指令

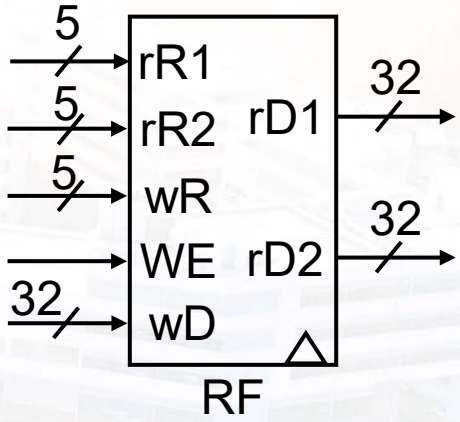
- 分解指令 —— 按照指令格式，分别取出各个字段
  - ✓ Tips：用**assign**语句、位选择符**[]**、位拼接符**{}**实现



ID单元实现 – 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

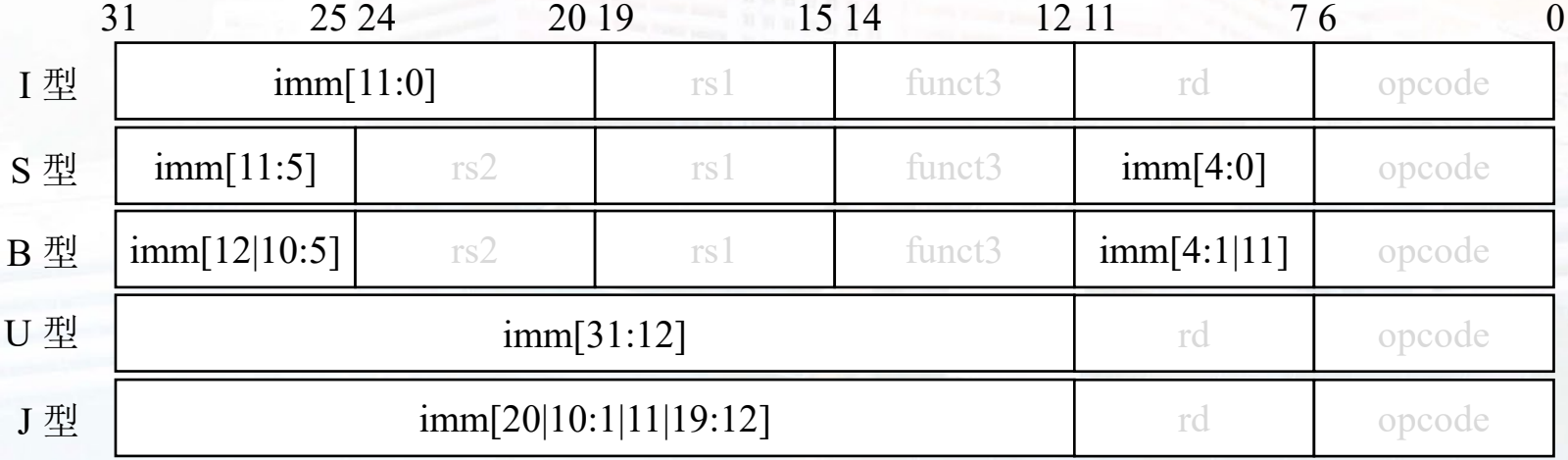
- RF的读写逻辑：
  - 异步读 (组合逻辑)、同步写 (时序逻辑)
  - WE=0: rD1<-REG[rR1]， rD2<-REG[rR2]
  - WE=1: REG[wR] <- wD



ID单元实现 – 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

miniLA: andi、ori、xori是零扩展



实验步骤

① 根据两个设计表，确定取指、译码单元**包含哪些部件**，以及这些部件的**接口**和需要实现的**功能**

② 下载模板工程，使用Verilog HDL实现各功能部件

③ 根据数据通路表，将各功能**部件连接**起来

④ **封装**成模块，得到取指单元和译码单元

- **注意代码规范性，可以避开很多问题！**

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

所画的图要和数据通路表对应！

开始实验