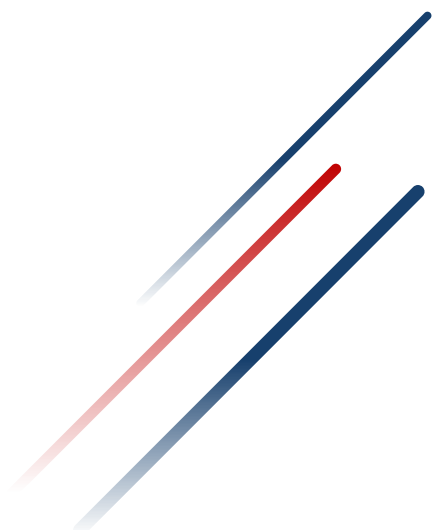


计算机设计与实践

单周期CPU设计-1



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



实验目的

- ◆ 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



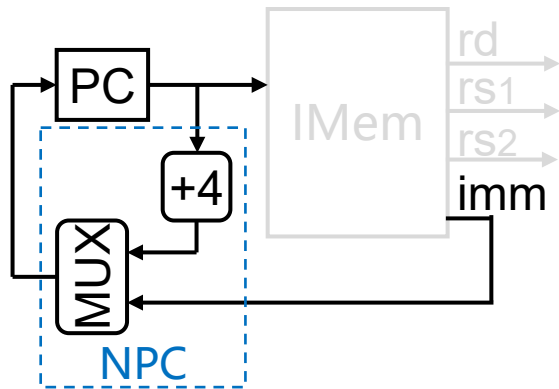
实验原理

- ◆ 根据数据通路表，确定取指、译码单元包含哪些部件
 - **取指单元**：PC、NPC、IROM (示例)
 - **译码单元**：RF、SEXT (示例)
- ◆ 首先实现取指、译码单元所需的功能部件



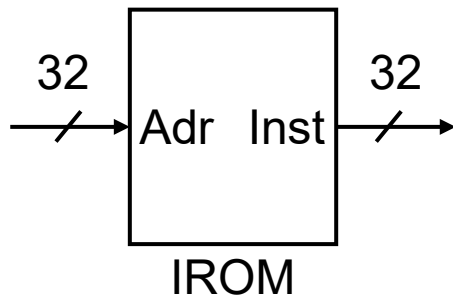
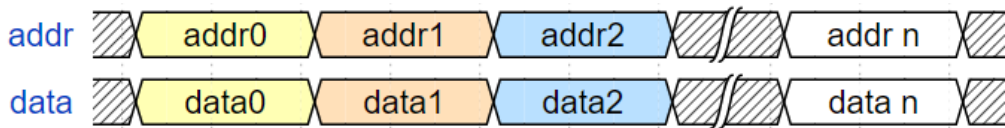
构建功能部件 —— PC和NPC

- ◆ 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: $PC+4$ 、 $PC+\text{sext}(\text{imm})$
 - ◆ 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：



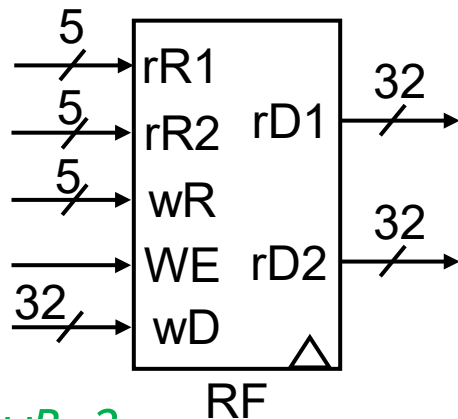
构建功能部件 —— RF

- ◆ 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:

- ◆ $WE=0: rD1 \leftarrow REG[rR1], \quad rD2 \leftarrow REG[rR2]$
- ◆ $WE=1: REG[wR] \leftarrow wD$

◆ **e.g.** `add x2, x9, x5` *# rR1=9, rR2=5; wR=2*



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

	31	25 24	20 19	15 14	12 11	7 6	0
I 型	imm[11:0]		rs1	funct3	rd	opcode	
S 型	imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode	
B 型	imm[12 10:5]	rs2	rs1	funct3	imm[4:1 11]	opcode	
U 型	imm[31:12]				rd	opcode	
J 型	imm[20 10:1 11 19:12]				rd	opcode	



取指单元实现

- ◆ 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



译码模块实现

- ◆ 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

	31	25 24	20 19	15 14	12 11	7 6	0
R 型	funct7		rs2	rs1	funct3	rd	opcode
I 型	imm[11:0]			rs1	funct3	rd	opcode
S 型	imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode
B 型	imm[12 10:5]		rs2	rs1	funct3	imm[4:1 11]	opcode
U 型	imm[31:12]					rd	opcode
J 型	imm[20 10:1 11 19:12]					rd	opcode



译码模块实现

- ◆ 根据指令格式，分解指令中的各个字段
- ◆ 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



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

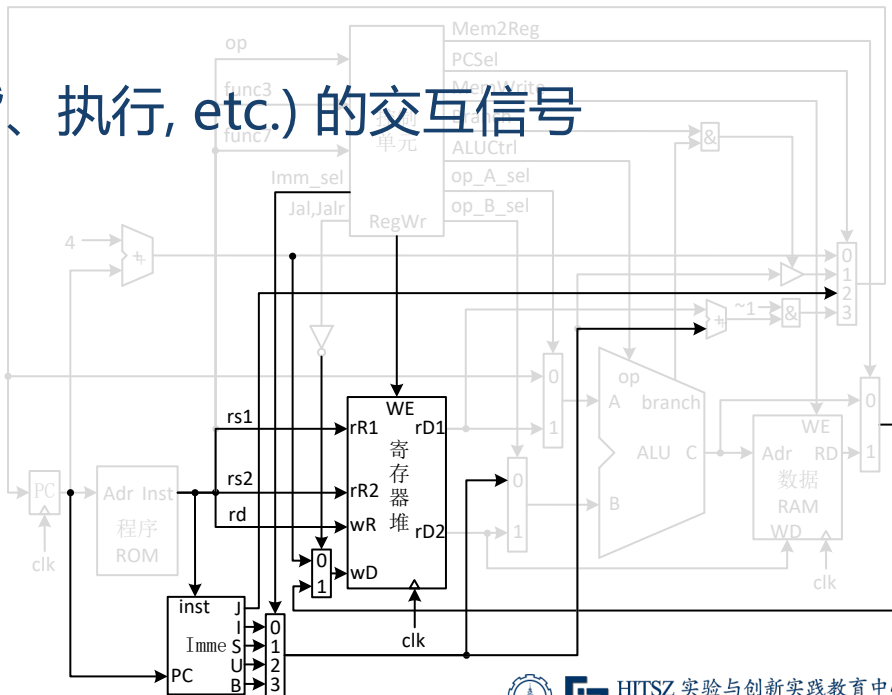


课堂检查

- ◆ 将自己设计的译码单元画出来 (2分)
- ◆ 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！





Let's roll !



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