版本

更新记录	文档名	实验指导书_lab2
	版本号	0.2
	创建人	计算机组成原理教学组
	创建日期	2022/1/1

更新历史

序号	更新日期	更新人	版本号	更新内容
1	2022/1/1	陈颖琪	0.1	单周期 CPU 取指译码实验
2	2022/1/1	陈颖琪	0.2	更新2.4节
3	2022/3/29	陈颖琪	0.3	修订2.3节对sext解释

文档错误反馈： 本文档经多版本迭代，但由于作者精力能力有限，其中出现错误请联系：

Yingqichen@sjtu.edu.cn

1 实验二 之存储器与控制器实验（单周期 CPU 取指译码）

本次实验开始涉及 RISC-V 架构 CPU 的设计，其中涵盖 CPU 在流水线设计中所分割的两个阶段，以下为实验概述：

RISC-V 架构 CPU 的传统流程可分为取指、译码、执行、访存、回写 (Instruction Fetch, Decode, Execution, Memory Request, Write Back)，五阶段。实验一完成了执行阶段的 ALU 部分，进行了访存实验，本实验将实现取指、译码两个阶段的功能。

在进行本次实验前，你需要具备以下实验环境及基础能力：

1. 了解 Xilinx Block Memory Generator IP 的使用
2. 了解数据通路、控制器的概念

1.1 实验目的

1. 掌握单周期CPU各个控制信号的作用和生成过程。
2. 掌握单周期CPU控制器的工作原理及其设计方法。
3. 理解单周期CPU执行指令的过程。
4. 掌握取指、译码阶段数据通路、控制器的执行过程。
5. 掌握RISC-V 架构下不同指令的立即数的构成和立即数产生模块的设计方法。

1.2 实验设备

1. 计算机 1 台 (尽可能达到 8G 及以上内存)；
2. EGO1实验开发板；
3. Xilinx Vivado 开发套件(2020.2 版本)。

1.3 存储器与控制器设计与验证实验任务

1.3.1 实验要求

图1为本次实验所需完成内容的原理图，依据取指、译码阶段的需求，需要在实验一基础上实现以下模块:

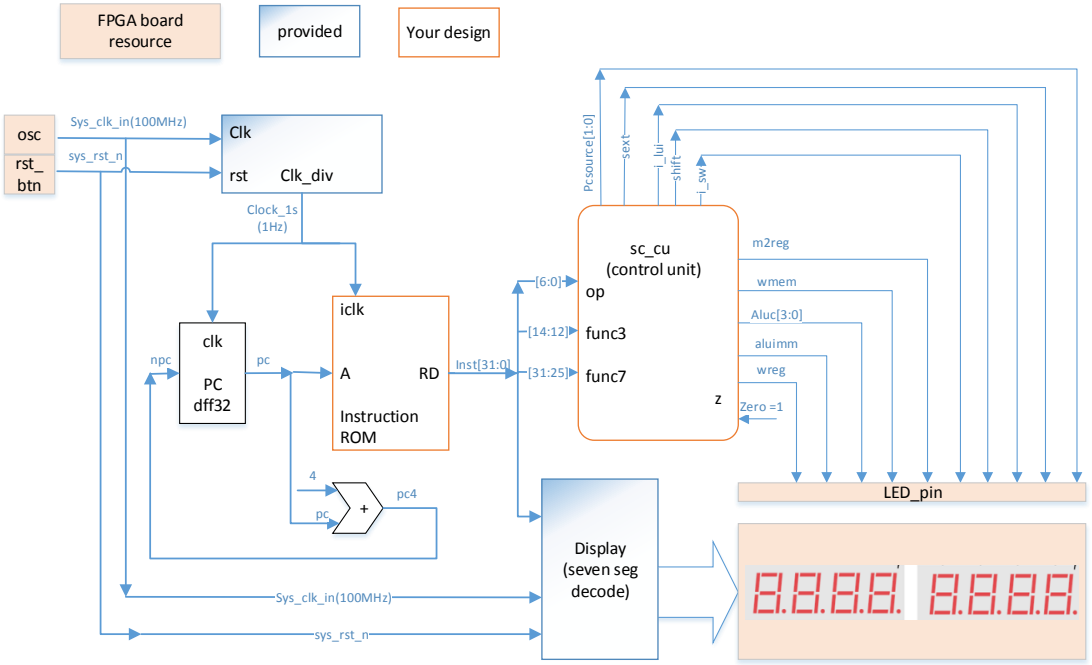

图 1: 取指译码原理图

1. PC。D 触发器结构，用于储存 PC(一个周期)。需实现 2 个输入，分别为 *clk*, *clrn*, 分别连接时钟和复位信号；需实现 1个输出*pc*, 连接指令存储器的地址A端口，*ena端口设置为始终允许*。其中A的位数N在例化指令存储器时指定，依据coe 文件中指令数，设定的存储器深度，必须足以存储所有指令。模块代码已提供；
2. 加法器。用于计算下一条指令地址，需实现 2 个输入，1 个输出，输入值分别为当前指令地址 *PC*、*32'h4*；
3. 控制单元。负责判断指令类型，并生成相应的控制信号。需实现1个输入，为指令inst，输出控制信号有多个，可作为多个输出，也可作为一个多位输出，具体参照 3进行设计；
4. 指令存储器。使用 Block Memory Generator IP 构造。(参考 2)

   注意：Basic 中 Generate address interface with 32 bits 选项不选中；PortA Options 中 Enable Port Type 选择为Use ENA Pin
5. 显示驱动。数码管七段译码显示驱动模块。（参考实验一的附件C，模块代码已提供）

6. 时钟分频器。将板载 100Mhz 频率降低为约1hz，连接PC、指令存储器时钟信号 clk。(已提供参考代码)
   注意：Xilinx Clocking Wizard IP 可分的最低频率为 4.687Mhz，因而只能使用自实现分频模块进行分频。

1.3.2 实验步骤

1. 从实验一和本实验已给代码中，导入 Display、seg7、clk_div、dff32模块
2. 使用Block Memory，导入lab2.2产生的 coe 文件，创建IROM模块
3. 创建控制器sc_cu模块
4. 导入提供的待补充顶层文件irom_test_top，连接相关模块
5. 导入顶层管脚约束文件sc_computer_iotest.xdc
6. 完成设计综合和FPGA实现，产生bit下载文件，编程下载到EGO1开发板
7. 完成板级验证

sc_cu输出信号与led 管脚对应关系如下表：

表 1: 输出信号与 led 管脚

zero	wmem	wreg	m2reg	aluc	aluimm	pcsource	sext	i_lui	i_sw	shift
[0:0]	[0:0]	[0:0]	[0:0]	[3:0]	[0:0]	[1:0]	[0:0]	[0:0]	[0:0]	[0:0]
led[14]	led[13]	led[12]	led[11]	led[10:7]	led[6]	led[5:4]	led[3]	led[2]	led[1]	led[0]

初始化 ROM 存储器中的内容，clk_div模块产生1s的时钟信号，控制PC模块和PC+4模块产生每秒加4的地址PC，将对应的指令存储器中内容读出来，并通过七段数码管驱动模块Display译码驱动后在EGO1板的数码管上显示。

注意板级测试实验前要仔细阅读EGO1的用户手册文件Ego1_UserManual_v2.2.pdf，及其他使用注意事项文件，了解其板载相关资源的工作原理、连接方式和应用注意事项。

1.3.3 实验环境

以下表格中红色部分需自行实现，黑色部分于实验发布包中提供。

表 2: 实验文件树

|--irom_test_top.v	设计顶层文件，参照图 1.3 将各模块连接。
|----clk_div.v	时钟分频1s，已提供
|----dff32.v	32bitD触发器，已提供
|----irom.ip	ROM IP，通过Block memory generator 进行实例化
|----sc_cu.v	控制器模块
|----irom.coe	RAM 初始化文件，已提供
|----display.v	七段数码管显示模块文件，已提供。
|------seg7.v	七段数码管显示模块组成文件，已提供。
|--sc_computer_iotest.xdc	综合实现时，约束文件，已提供。

1.4 立即数产生模块实验任务

1.4.1 任务

RISC-V架构指令中含有立即数操作的不同类型指令的机器码中立即数构成方式不尽相同。本模块用于区分支持不同类型的指令时，产生所需要的经有符号扩展后的32bit立即数，用于后续ALU模块的计算或者对下一条指令的PC值更新。

参阅本文2.4节原理讲解完成immext模块的verilog语言程序设计。

1.4.2 步骤

1、在实验包提供的excel文件“lab2.3 补全sc_cu真值表_RISCV_to_student.xlsx”中，列出了待实现的22种指令，其中I、S、SB、U、UJ型指令会用到立即数。分析并完善下列表格中各个列的内容，协助理解设计原理。

2、完成本模块的verilog语言程序设计。

输入		**(依赖于 sext，i_lui,i_sw,shift，pcsource)**	
指令类型	**ext_imm[31:0] imm 组成**	**ext_imm[31:0] inst 来源组成**	**PC(依赖于 pcsource)**
addi	itype_imm {20{imm[11]}, imm[11:0]}	{20{inst[31]}, inst[31:20]}	PC=PC+4
andi			
ori			
xori			
slli			
srli			
srai			
lw			
jalr			
sw			
beq			
bne			
lui			
jal			

2 实验原理

2.1 取指阶段原理

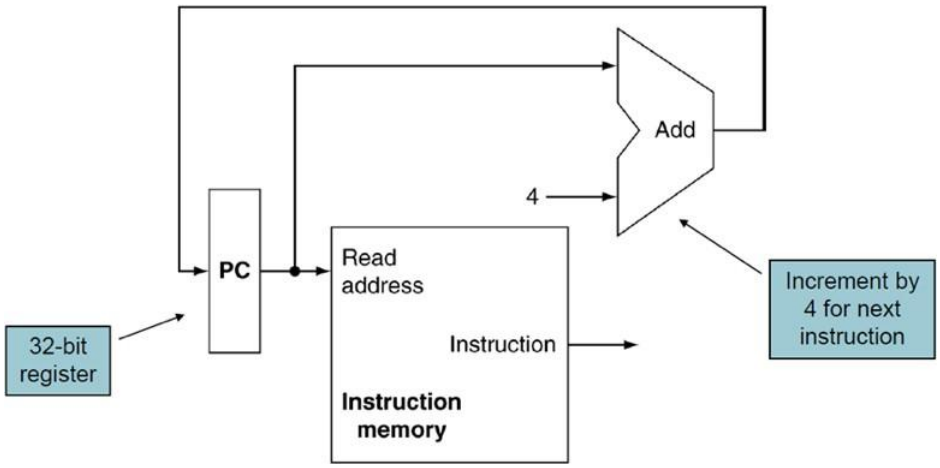

图 8: 取指示意图

如图 8所示，PC 为 32bit(1 word) 的寄存器，存放指令地址，每条指令执行完毕后，增加 4，即为下一条指令存放地址。指令地址传入指令存储器，即可取出相应地址存放的指令。

需要注意的是，RISC-V架构中，采用字节读写，1 32bit word = 4 byte，故需要地址 +4 来获取下一条指令。

2.2 指令译码原理

CORE INSTRUCTION FORMATS

	31 27	26 25	24 20	19 15	14 12	11 7	6 0
R	funct7		rs2	rs1	funct3	rd	opcode
I	imm[11:0]			rs1	funct3	rd	opcode
S	imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode
SB	imm[12|10:5]		rs2	rs1	funct3	imm[4:1|11]	opcode
U	imm[31:12]					rd	opcode
UJ	imm[20|10:1|11|19:12]					rd	opcode

图 9: RISC-V 指令原理图

如图 9所示，32 位 RISC-V 指令在不同类型指令中分别有不同结构。但 [6:0] 表示的 opcode，以及 [31:25] 表示的 funct7，[14:12] 表示的 funct3，为译码阶段明确指令控制信号的主要字段。表 3为 opcode 及 funct7，funct3 识别得到的部分信号，详细信号表参照课本和参考资料 RISC-V-Reference-Data.pdf，及文件 lab2.3 补全sc_cu真值表_RISCV_to_student.xlsx第二页。

表 3: 译码控制信号

opcode	operation	Opcode	Funct7	Funct3	alu function	aluc
lw	Load word	0000011	XXXXXX	010	add	0000
sw	Store word	0100011	XXXXXX	010	add	0000
beq	Branch equal	1100011	XXXXXX	000	subtact	1000
R-type	add	0110011	0000000	000	add	0000
	subtract	0110011	0100000	000	subtract	1000
	and	0110011	0000000	111	and	0111
	or	0110011	0000000	110	or	0110

2.3 控制器实现原理

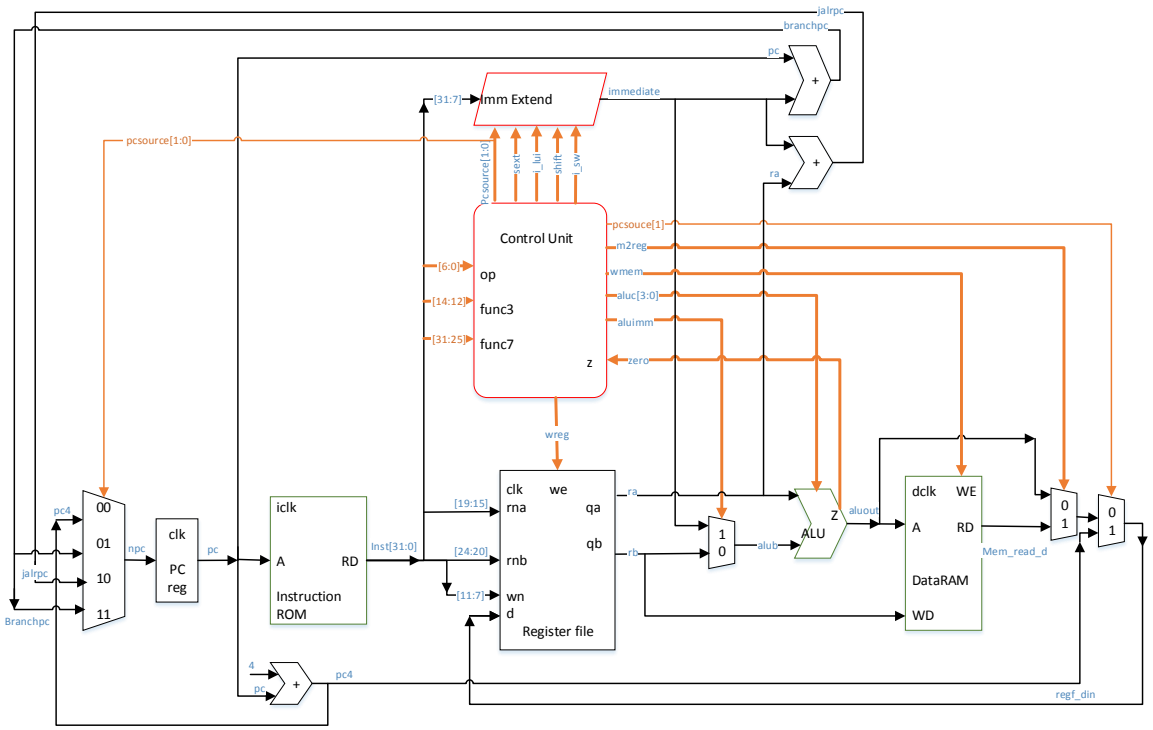

图 10: 22条指令单周期数据通路

由图10可知，控制器输出的控制信号，用于控制数据通路器件的使能、多路选择器的选择，以及产生扩展立即数。因此，根据不同指令的功能分析其所需要的路径，即可得到信号所对应的值。在此之前，参照表4对各个控制信号的含义进行理解。

表 4: 信号含义

信号	含义
wmem	数据存储器写允许是否需要写数据存储器
wreg	regfile 写允许
m2reg	lw指令，选择来自存储器数据写入regfile
aluc	alu功能控制，4bit最多可支持16种alu操作。具体定义主要来自{inst[30],func3[2:0]}，满足区分22种指令的操作，且能简化合并可有相同操作的指令
aluimm	alub输入端选择立即数进行运算,而非regfile的b输出端
pcsource	2bit，选择更新的pc指针来自哪里。pcsource[1]: 也用于jal、jalr指令时设置rd=PC+4
sext	符号扩展标志
i_lui	lui指令标志
i_sw	sw指令标志
shift	移位操作指令标志

理解指令执行过程，分析数据通路图，判断指令是否需要写寄存器、访存等等操作，以产生相应的控制信号。

以 add 指令为例，add rd, rs1, rs2，要完成将 reg[rs1] + reg[rs2] 的和写入rd寄存器的功能。对应add指令，控制信号应该设为：

Pcsource设置为00，表示下一条指令从PC+4取指令；

Aluc应该设置为0000，表示进行加运算；

aluimm设置为0，表示该指令不含立即数操作，MUX选择寄存器堆的rb输出信号，连接到alub输入端；

wmem设置为0，不需要向数据存储器写入数据，求和结果直接连接到后续MUX，用于写回寄存器rd；

wreg设置为1，表示该指令是要写rd寄存器的；

m2reg设置为0，表示MUX选择写回rd寄存器的数据是来自aluout，而非来自数据存储器的读出数据mem_resd_d，（图10中最右侧MUX用到控制信号pcsource[1]，表示用于jal、jalr指令时设置rd=PC+4，其他指令时选择来自前一个MUX的输出值）；

另外四个信号是用于控制立即数产生模块信号的：

shift设置为0，表示该指令不是移位指令；

sext设置为0，表示该指令不需要做立即数有符号扩展操作；

i_lui设置为0，为不是lui指令（i_lui是lui指令标志，只有lui指令时才设置为1）；

i_sw设置为0，为不是sw指令（i_sw是sw指令标志，只有sw指令时才设置为1）；

表中只给出了部分指令的控制信号输出设置，请补充完整表5（也可在实验包提供的excel文件lab2.3 补全sc_cu真值表_RISCV_to_student.xlsx中完成）。

表 5: 控制信号译码

输入 inst[31:0]			sc_cn 输出									
指令类型	指令	ALU的输出 z	pcsource [1..0]	aluc [3..0]	aluimm	shift	sext	wmem	wreg	m2reg	i_lui	i_sw
R	add	x	0 0	0000	0	0	0	0	1	0	0	0
	sub	x	0 0	1000	0	0	0	0	1	0	0	0
	and			0111								
	or			0110								
	xor			0100								

	sll	x	0 0	0001	0	1	0	0	1	0	0	0
	srl			0101								
	sra			1101								
I	addi	x	0 0	0000	1	0	1	0	1	0	0	0
	andi	x	0 0	0111	1	0	1	0	1	0	0	0
	ori											
	xori											
	slli	x	0 0	0001	1	1	1	0	1	0	0	0
	srli											
	srai											
	lw	x	0 0	0000	1	0	1	0	1	1	0	0
	jalr		1 0 (jalrpc)	0000								
S	sw											
SB	beq	0	0 0	1000	0	0	1	0	0	0	0	0
		1	0 1(bpc)									
	bne	0										
		1										
U	lui			0010								
UJ	jal	x	11(jpc)	xxxx	0	0	1	0	1	0	0	0

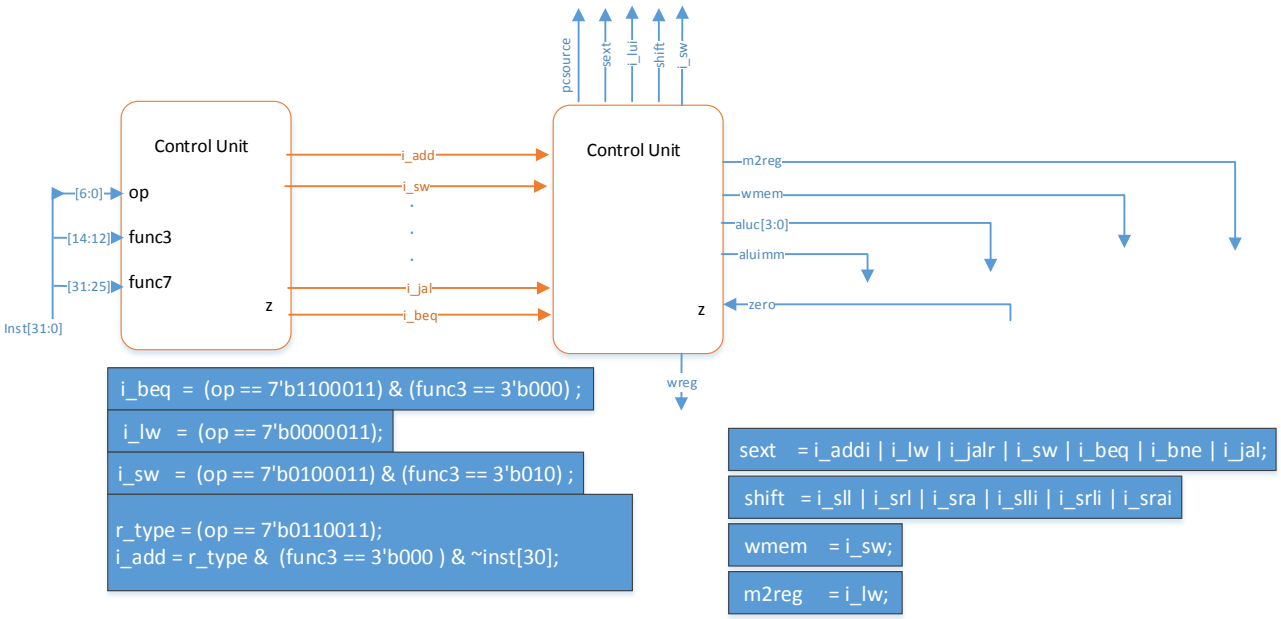

图11

经过分析得到完整的表5后，就可以完成控制器的设计。如图11 所示，可以分两步产生最终输出的控制信号。首先根据输入的指令inst信号，可以提取其中的op，func3，func7信息，产生各个指令标志，如图中示例的beq指令标志i_beq，有：i_beq = (op == 7'b1100011) & (func3 == 3'b000) ; lw指令的i_lw标志，sw指令的i_sw标志见图11。其他指令标志位请自行设计完成。

之后根据表5中的每个输出信号列，将所有该列中为1的标志选上，进行或操作，即为该信号有效标志。例如sext信号应该在i-type所有指令以及addi、lw、jalr、sw、beq、bne、jal指令中任意一个有效时为1，这时应该对指令中的立即数进行有符号扩展。其他的指令时不需要做立即数符号扩展。因此将对应指令标志相或即可。有 sext = i_addi | i_lw | i_jalr | i_sw | i_beq | i_bne | i_jal | i-type;（思考：图11中sext表达式完整吗？）再比如像wmem信号就只在sw指令时需要有效，以控制写入数据存储器。请依据表5完成所有控制信号逻辑描述。

2.4 立即数产生模块设计原理

立即数产生模块完成对不同类型含立即数的指令里的立即数做带符号扩展，构成32bit立即数。可以采用多路选择器，根据sc_cu模块来的控制信号 sext，i_lui, i_sw, shift，pcsource[1:0]，依据不同类型的指令，将对应类型指令机器码 inst[31:0] 中的立即数信息位，按照该指令定义重新排列组合，并考虑符号扩展后输出，形成32bit的 immediate 信号。也就是根据指令类型不同完成译码操作。图12中，e为某个指令经是否需要扩展标志标识后的扩展位的值。可以按照如图12中的从上到下的顺序描述不同指令下的immediate信号输出取值。

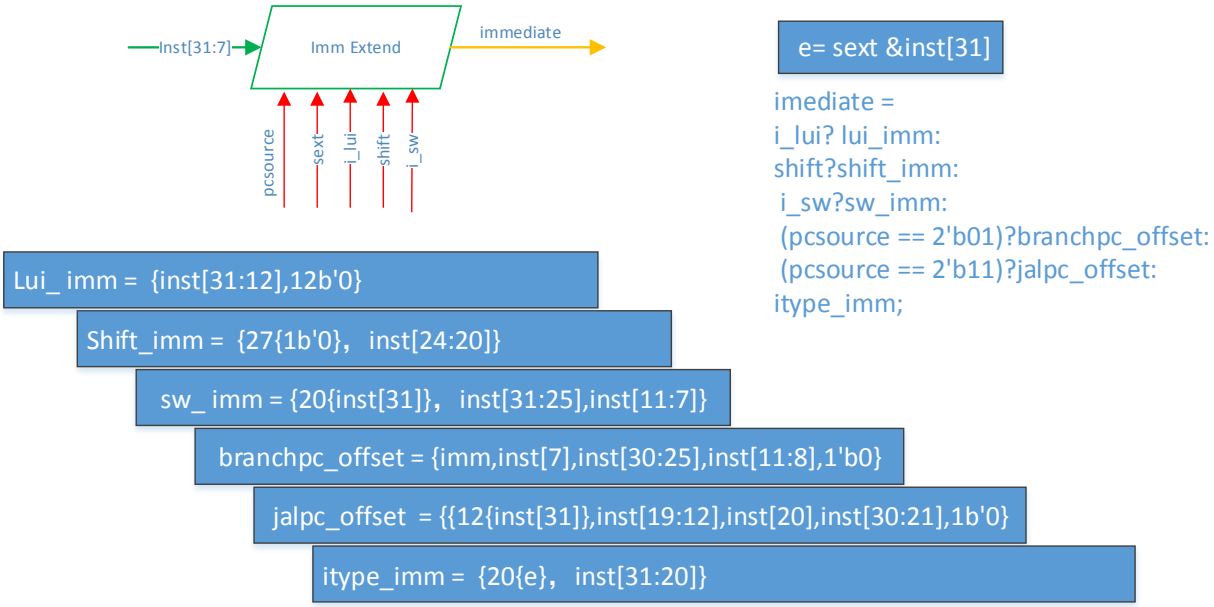

图 12

该模块接口可如下定义：

```
module immext(inst,pcsource,sext,i_lui,i_sw,shift,out_immediate);
input [31:0] inst;
input sext,i_lui,i_sw,shift;
input [1:0] pcsource;
output [31:0] out_immediate;
```

根据图12补充完整,存为immext.v。

思考：图12中的写法是否有纰漏？构成branchpc_offset的信号imm应该怎么定义？构成sw_imm的高20位用20{e}代替20{inst[31]}，构成shift_imm的高27位用27{e}代替27{1b’0}，构成jalpc_imm的高12位用12{e}代替12{inst[31]}是不是更合理？

3 实验报告要求

实验报告应简明扼要，重点突出。具体内容包括以下几部分：

1. 实验目的。
2. 本次实验所用的实验平台、仪器以及其它实验器材和部件等。
3. 实验任务。这部分是实验报告最主要的内容。根据任务要求，描述设计原理与思路，设计的实现方式，结构框架等。（不要仅贴整段Verilog 程序代码，而是以表述介绍设计思路为主，如有必要，可贴关键代码段，并加注释）。结合实验原理，描述设计的全过程和实验步骤。
4. 实验结果演示总结。板上验证结果可以有视频照片演示的截图，仿真结果可以有仿真波形关键部分的截图和解释，不要只贴截图波形。必须有对应的解释，指出该部分截图要说明什么问题。
5. 整理、归纳做完的实验内容并解释相应的实验现象或问题。最后也可总结心得体会。