

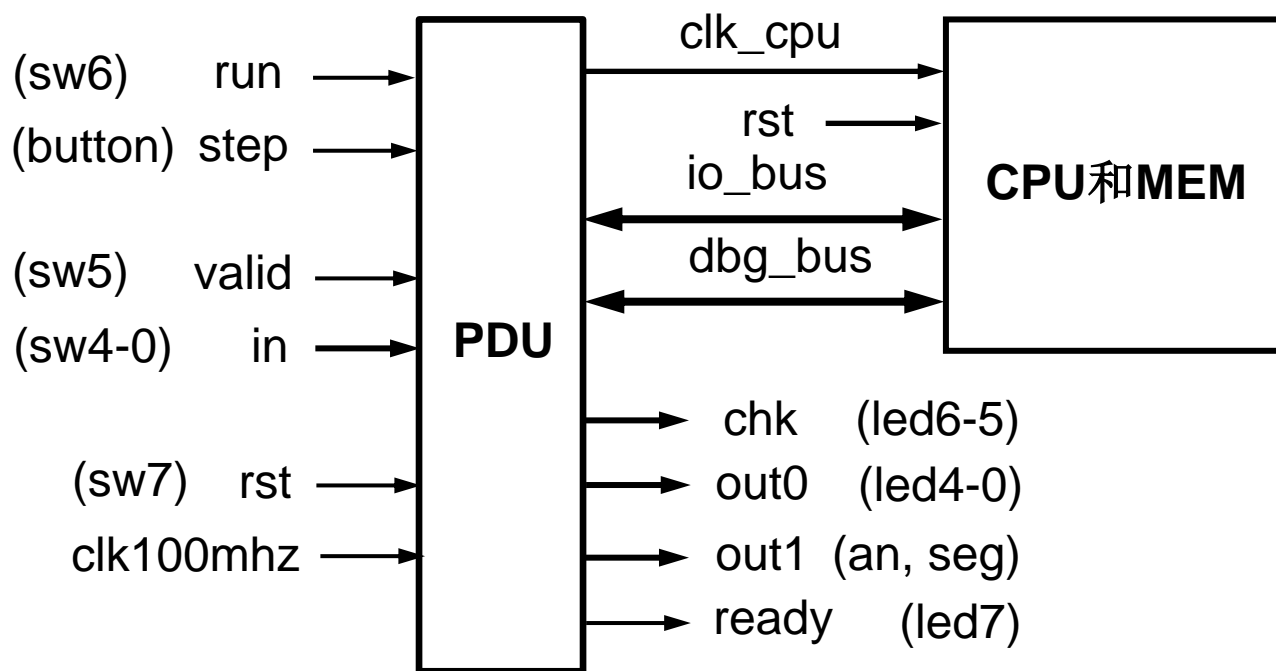
实验四 单周期CPU设计

实验目标

- 理解CPU的结构和工作原理
- 掌握单周期CPU的设计和调试方法
- 熟练掌握数据通路和控制器的设计和描述方法

实验内容

- 设计实现单周期RISC-V CPU，具有如下6条指令：
 - add, addi, lw, sw, beq, jal



运算指令

- `add rd, rs1, rs2` $\# x[rd] = x[rs1] + x[rs2]$

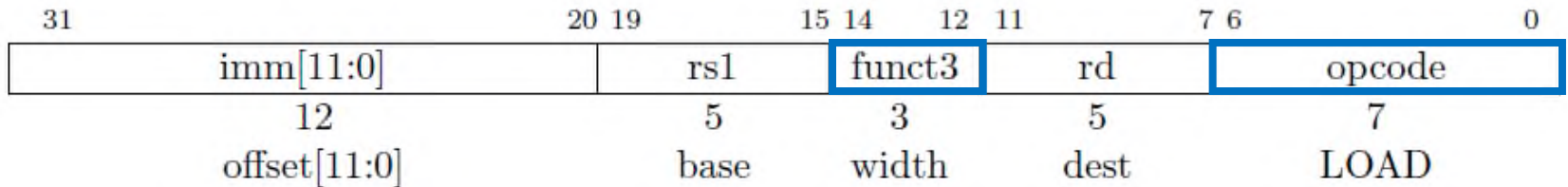
31	25 24	20 19	15 14	12 11	7 6	0
funct7	rs2	rs1	funct3	rd	opcode	
7	5	5	3	5	7	
0000000	src2	src1	ADD/SLT/SLTU	dest	OP	
0000000	src2	src1	AND/OR/XOR	dest	OP	
0000000	src2	src1	SLL/SRL	dest	OP	
0100000	src2	src1	SUB/SRA	dest	OP	

- `addi rd, rs1, imm` $\# x[rd] = x[rs1] + sext(imm)$

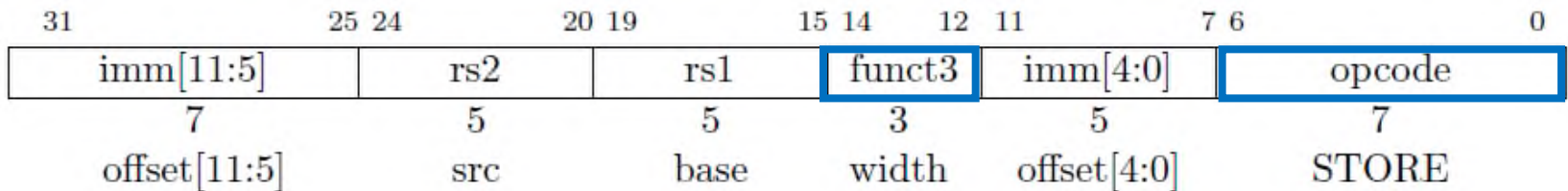
31	20 19	15 14	12 11	7 6	0
imm[11:0]	rs1	funct3	rd	opcode	
12	5	3	5	7	
I-immediate[11:0]	src	ADDI/SLTI[U]	dest	OP-IMM	
I-immediate[11:0]	src	ANDI/ORI/XORI	dest	OP-IMM	

访存指令

- `lw rd, offset(rs1)` $\# x[rd] = M[x[rs1] + sext(offset)]$

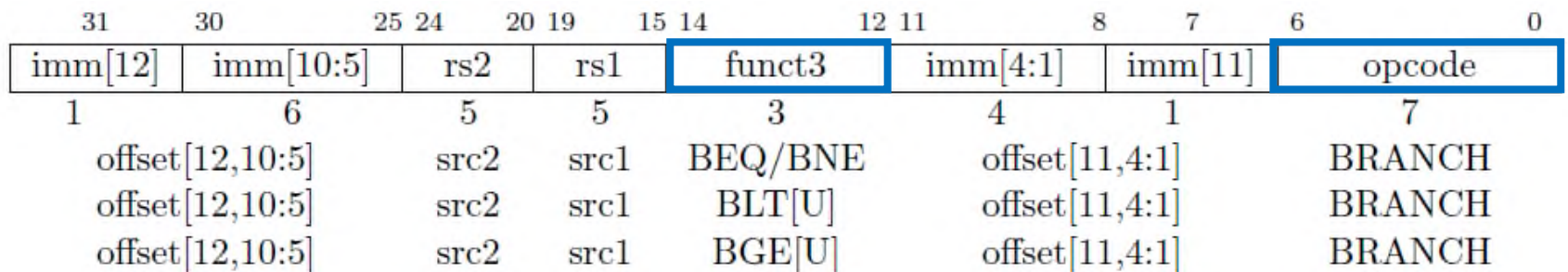


- `sw rs2, offset(rs1)` $\# M[x[rs1] + sext(offset)] = x[rs2]$

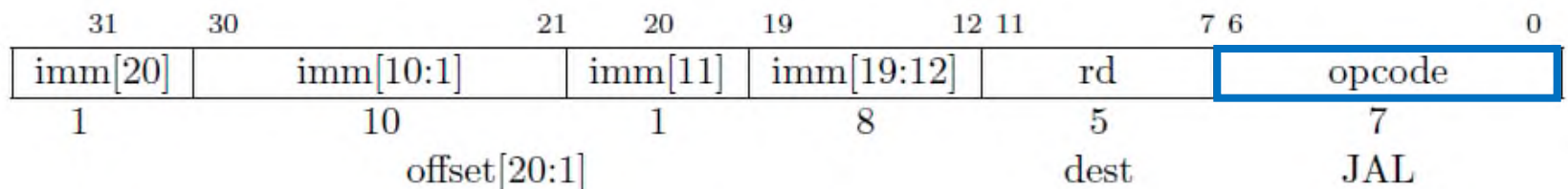


分支和跳转指令

- `beq rs1, rs2, offset` # if (rs1 == rs2) pc += sext(offset)



- `jal rd, offset` # $x[rd] = pc+4$; pc += sext(offset)



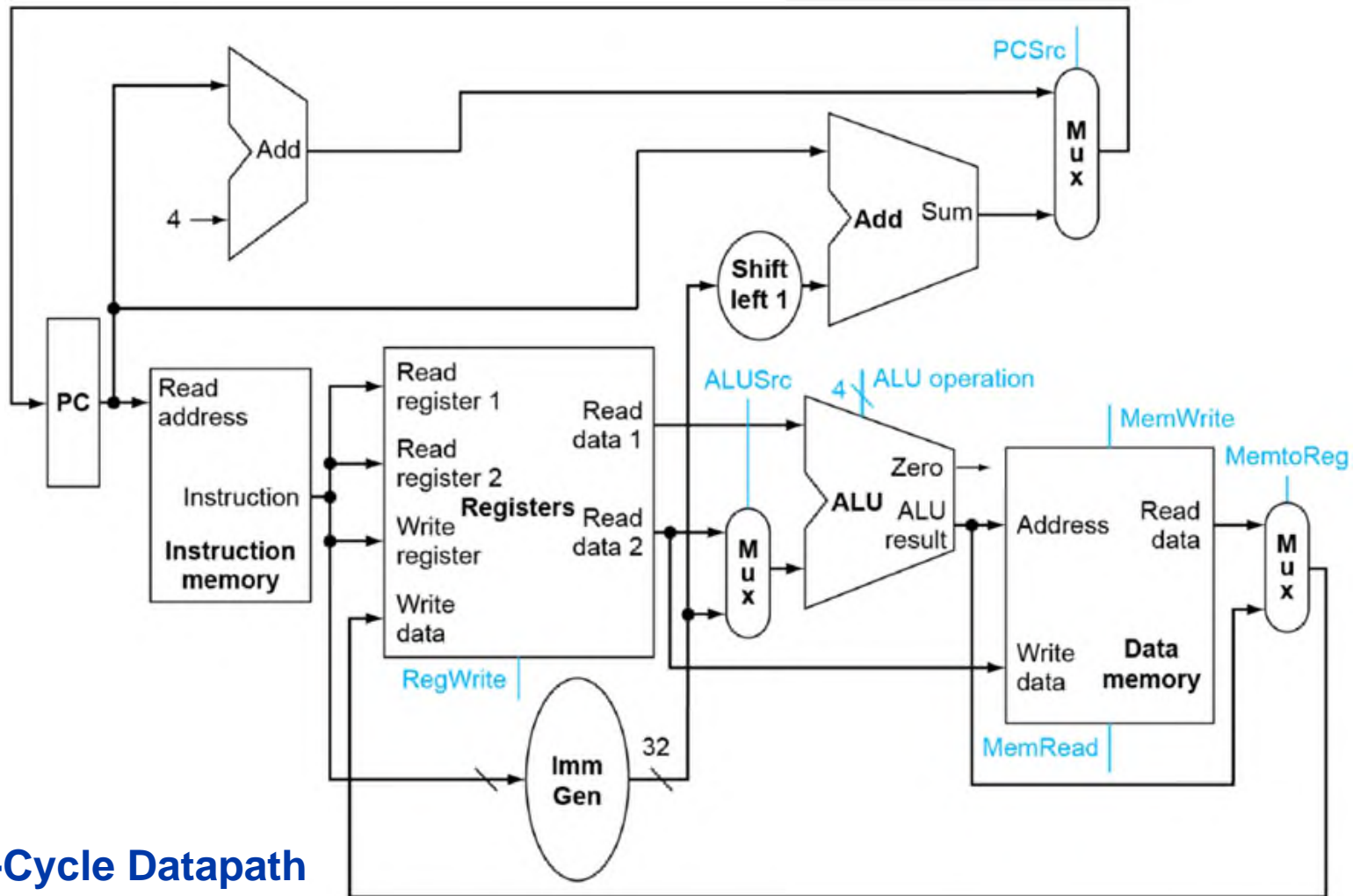
RV32I 指令编码

imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20 10:1 11 19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12 10:5]	rs2	rs1	000	imm[4:1 11]	1100011	BEQ
imm[12 10:5]	rs2	rs1	001	imm[4:1 11]	1100011	BNE
imm[12 10:5]	rs2	rs1	100	imm[4:1 11]	1100011	BLT
imm[12 10:5]	rs2	rs1	101	imm[4:1 11]	1100011	BGE
imm[12 10:5]	rs2	rs1	110	imm[4:1 11]	1100011	BLTU
imm[12 10:5]	rs2	rs1	111	imm[4:1 11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW

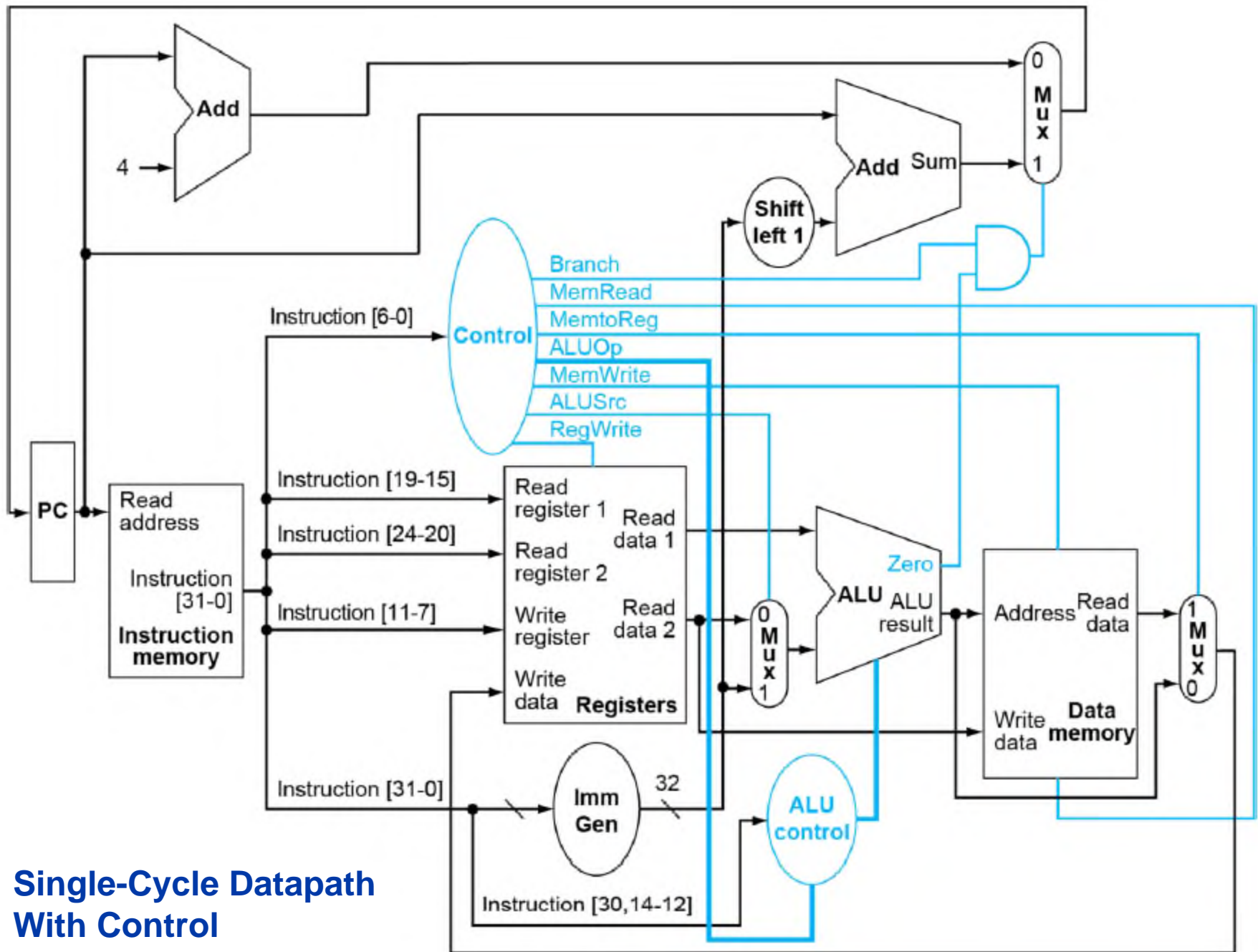
RV32I 指令编码 (续)

imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND

单周期CPU数据通路



Single-Cycle Datapath

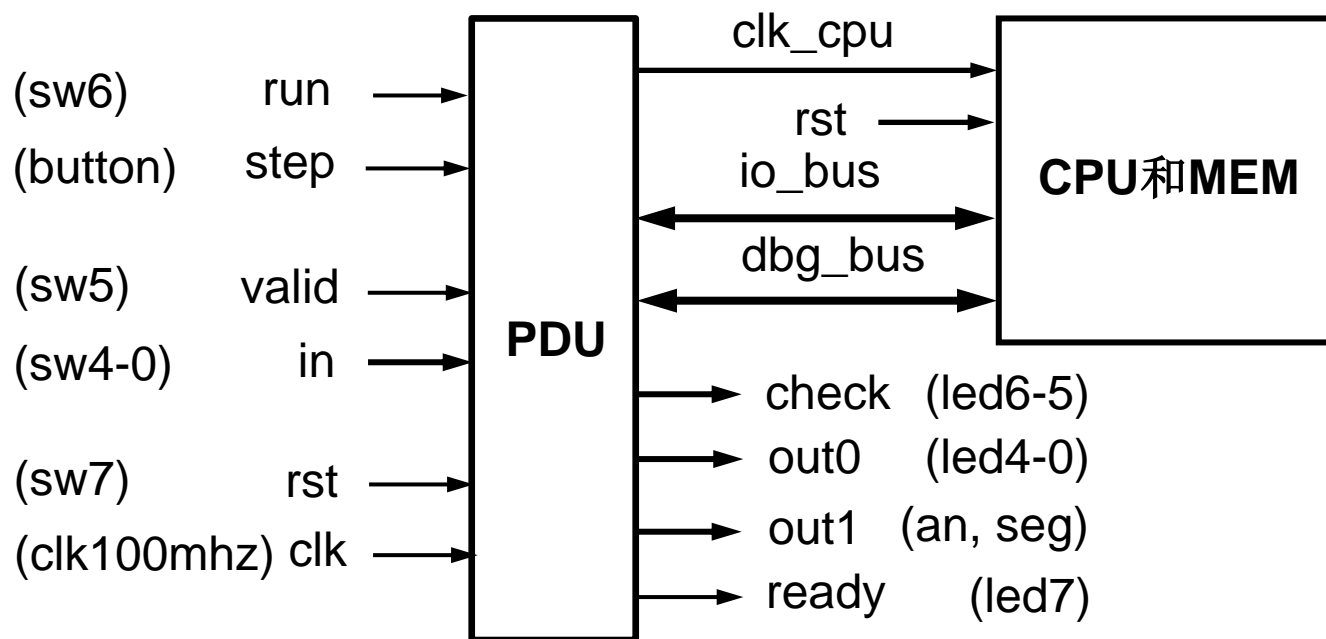


**Single-Cycle Datapath
With Control**

处理器调试单元

- **PDU (Processor Debug Unit)**

- 控制CPU的运行方式: **run = 1** 连续运行, **0** 单步运行
- 管理外设 (开关sw、指示灯led、数码管an & seg), 显示运行结果和数据通路状态



IO_BUS信号

- CPU运行时访问开关(sw)、指示灯(led)和数码管(an, seg)
 - io_addr: I/O外设的地址
 - io_din: CPU接收来自输入缓冲寄存器 (IBR) 的sw输入数据
 - io_dout: CPU向led和seg输出的数据
 - io_we: CPU向led和seg输出时的使能信号, 利用该信号将io_dout存入输出缓冲寄存器 (OBR), 再经数码管显示电路将其显示在数码管 (an, seg)

DBG_BUS信号

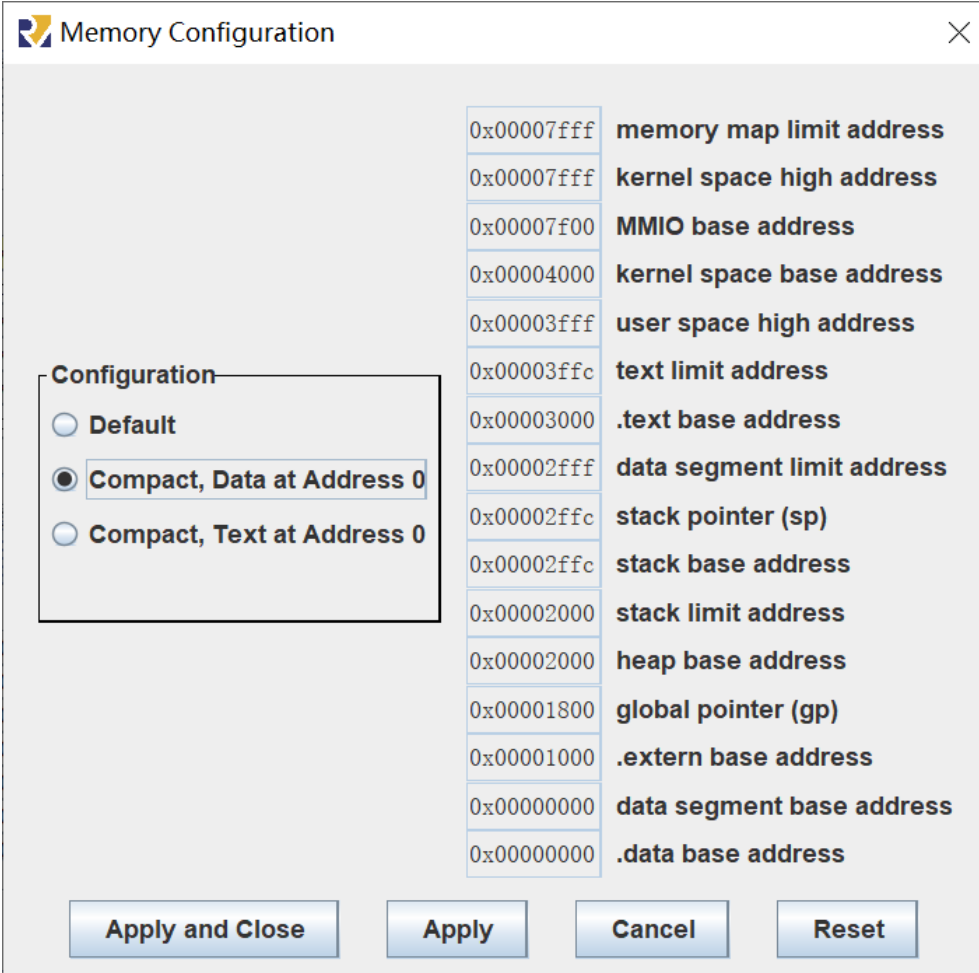
- 调试时将存储器和寄存器堆内容，以及CPU数据通路状态信息导出显示
 - m_rf_addr: 存储器(MEM)或寄存器堆(RF)的调试读口地址
 - rf_data: 从RF读取的数据
 - m_data: 从MEM读取的数据
 - pc: PC的内容

CPU运行方式

- **run = 1: 连续运行**
 - PDU向CPU输出连续时钟信号clk_cpu
 - CPU通过I/O_BUS访问外设
 - 输入端口: in, valid
 - 输出端口: out0, out1, ready
- **run = 0: 单步运行（每次执行一条指令）**
 - 每按动step一次, PDU产生一个周期的clk_cpu
 - 执行外设输入指令前, 应先设置好valid或in后, 再按动step
 - 执行任何指令后, led和数码管(an, seg)显示当前程序运行结果
 - 随后可以通过改变valid和in查看寄存器堆、存储器和PC的内容

RARS存储器配置

- 紧凑且数据地址为0
 - 0x0000_0000 ~ 0x0000_2ffff
- 代码地址:
 - 0x0000_3000 ~ 0x0000_3ffc



The image shows a 'Memory Configuration' dialog box with a 'Configuration' section containing three radio buttons: 'Default', 'Compact, Data at Address 0' (which is selected), and 'Compact, Text at Address 0'. To the right of the configuration section is a list of memory addresses and their corresponding labels. At the bottom of the dialog are four buttons: 'Apply and Close', 'Apply', 'Cancel', and 'Reset'.

Address	Label
0x00007fff	memory map limit address
0x00007fff	kernel space high address
0x00007f00	MMIO base address
0x00004000	kernel space base address
0x00003fff	user space high address
0x00003ffc	text limit address
0x00003000	.text base address
0x00002fff	data segment limit address
0x00002ffc	stack pointer (sp)
0x00002ffc	stack base address
0x00002000	stack limit address
0x00002000	heap base address
0x00001800	global pointer (gp)
0x00001000	.extern base address
0x00000000	data segment base address
0x00000000	.data base address

实验存储器配置

- 指令存储器(256x32bit)地址: **0x0000_3000 ~ 0x0000_33ff**
- 数据存储器(256x32bit)地址: **0x0000_0000 ~ 0x0000_03ff**
- 外设端口地址: **0x0000_0400 ~ 0x0000_07ff**

存储器映射外设端口地址表

存储器地址	I/O_addr	输出端口名	输入端口名	外设
0x0000_0400		out0	-	led4-0
0x0000_0404		ready	-	led7
0x0000_0408		out1	-	an, seg
0x0000_040C		-	in	sw4-0
0x0000_0410		-	valid	sw5

外设使用说明表

sw				button	led			an/seg	说明
7	6	5	4~0		7	6~5	4~0		
rst	run	valid	in	step	ready	check	out0	out1	
↑	-	-	-	-	1	00	0x1f	0x12...8	复位
x	1	valid	in	-	ready	00	out0	out1	连续运行
	0	valid	in	↑	ready	00	out0	out1	单步运行
		↑↓	addr_rf	x	0	01	addr_rf	data_rf	查看寄存器堆
			addr_m		0	10	addr_m	data_m	查看存储器
			-		0	11	0	PC	查看PC

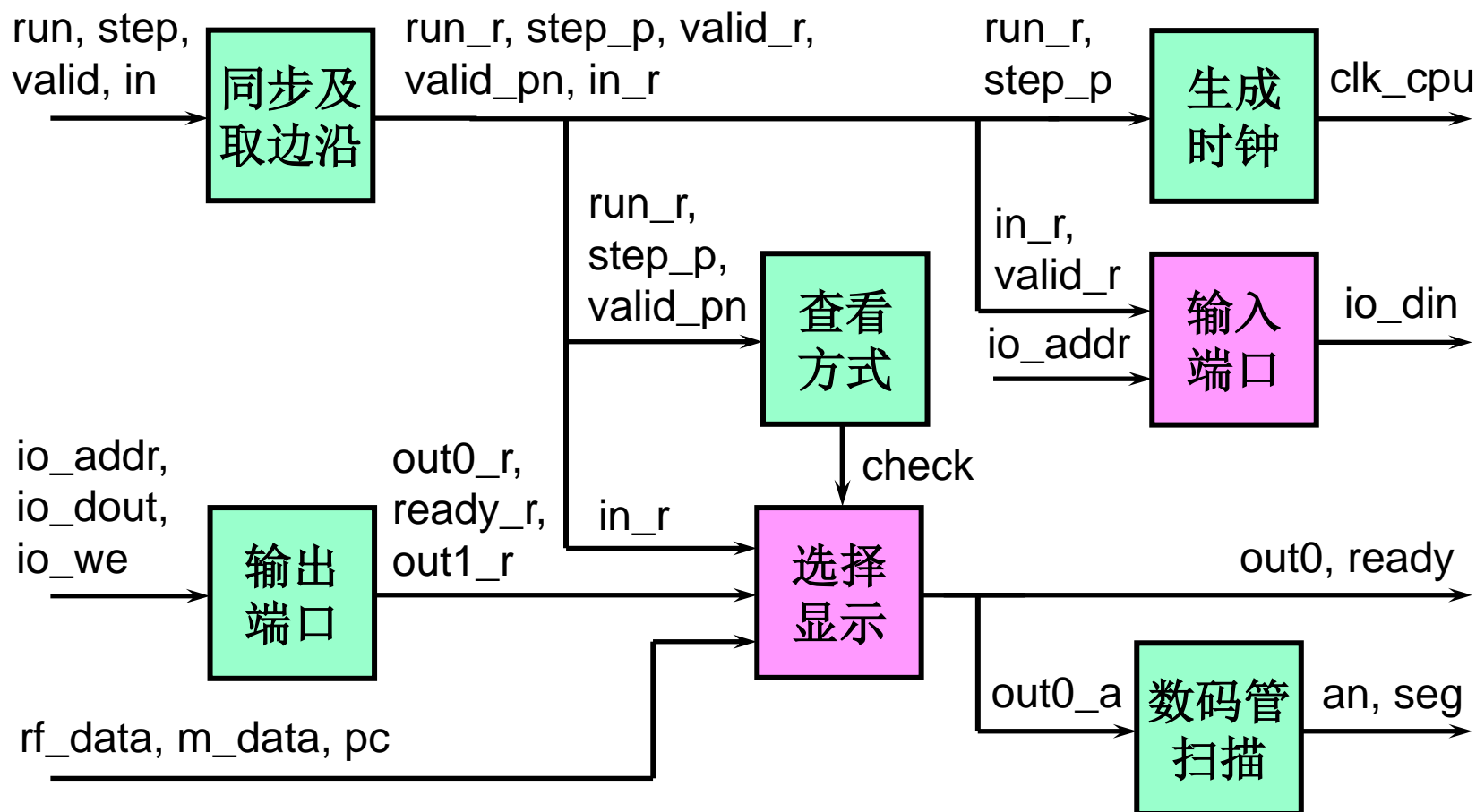
CPU模块接口

```
module cpu (  
    input clk,  
    input rst,  
  
    //IO_BUS  
    output [7:0] io_addr,    //led和seg的地址  
    output [31:0] io_dout,   //输出led和seg的数据  
    output io_we,           //输出led和seg数据时的使能信号  
    input [31:0] io_din,     //来自sw的输入数据  
  
    //Debug_BUS  
    input [7:0] m_rf_addr,   //存储器(MEM)或寄存器堆(RF)的调试读口地址  
    output [31:0] rf_data,   //从RF读取的数据  
    output [31:0] m_data,    //从MEM读取的数据  
    output [31:0] pc         //PC的内容  
);
```

PDU模块接口

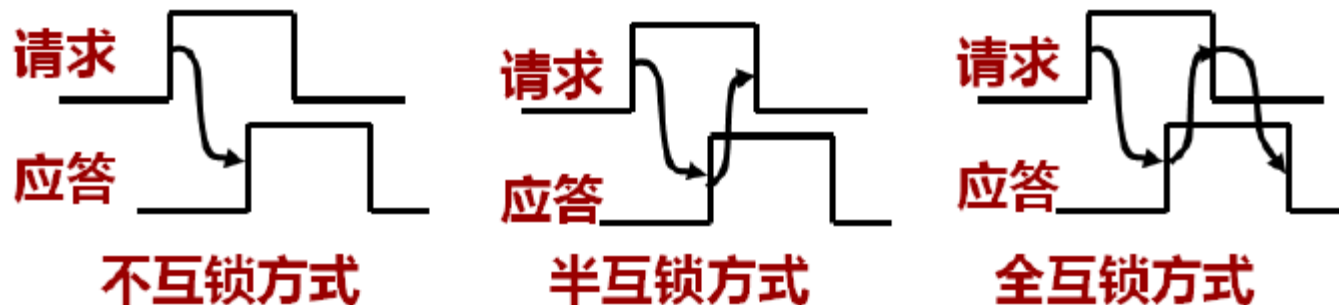
<pre>module pdu (input clk, input rst, //选择CPU工作方式 input run, input step, output clk_cpu, //输入sw的端口 input valid, input [4:0] in, //输出led和seg的端口 output [1:0] check, //led6-5:查看类型 output [4:0] out0, //led4-0</pre>	<pre> output [2:0] an, //8个数码管 output [3:0] seg, output ready, //led7 //IO_BUS input [7:0] io_addr, input [31:0] io_dout, input io_we, output [31:0] io_din, //Debug_BUS output [7:0] m_rf_addr, input [31:0] rf_data, input [31:0] m_data, input [31:0] pc);</pre>
--	---

PDU逻辑结构图

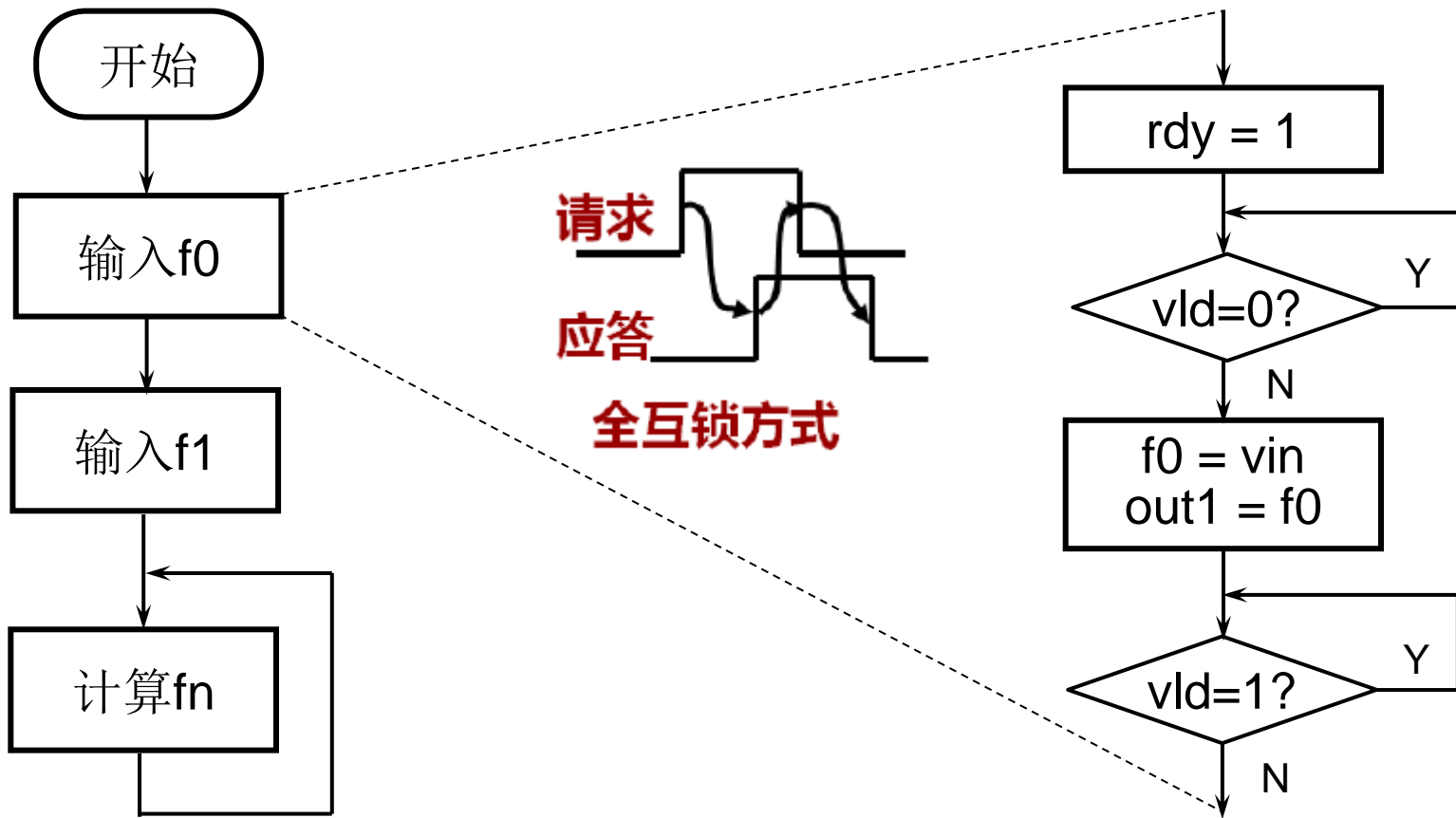


应用程序测试

- 设计汇编程序：计算斐波那契—卢卡斯数列
 - 依次输入数列开始两项（设置in后按动valid确认），并输出至数码管（out1）上显示
 - 计算数列后续项，并输出至数码管上显示，按动valid继续
- 通过握手信号，协调CPU和外设的数据传输过程
 - ready: CPU准备好，指示外设提供数据
 - valid: 外设指示数据（in）有效



应用程序流程图



实验步骤

1. 修改Lab2寄存器堆模块，增加1个用于调试的读端口，且使其r0内容恒定为0
2. 结构化描述单周期CPU，并进行功能仿真
 - 假定指令存储器和数据存储器（增加一个读端口用于调试）均使用分布式存储器，容量均为256x32位，使用Lab3实验内容2生成的COE文件初始化
3. 将CPU和PDU下载至FPGA中测试，使用Lab3实验内容3生成的COE文件对指令存储器和数据存储器初始化

The End