浙江大學
ZHEJIANG UNIVERSITY

Imagination

“计算机组成与设计”课程实践
基于RVfpga的量化研究

浙江大学 – **刘鹏**

课程框架

浙江大学 ZHEJIANG UNIVERSITY

实验设计	加速指令设计	指令实现与验证	存储与编译优化
知识范围	RISC-V指令设计	指令实现与验证	存储系统
基础支撑	RVfpga课程	课题组科研	
教学理念	科研教学融合	理论实验融合	量化计算思维

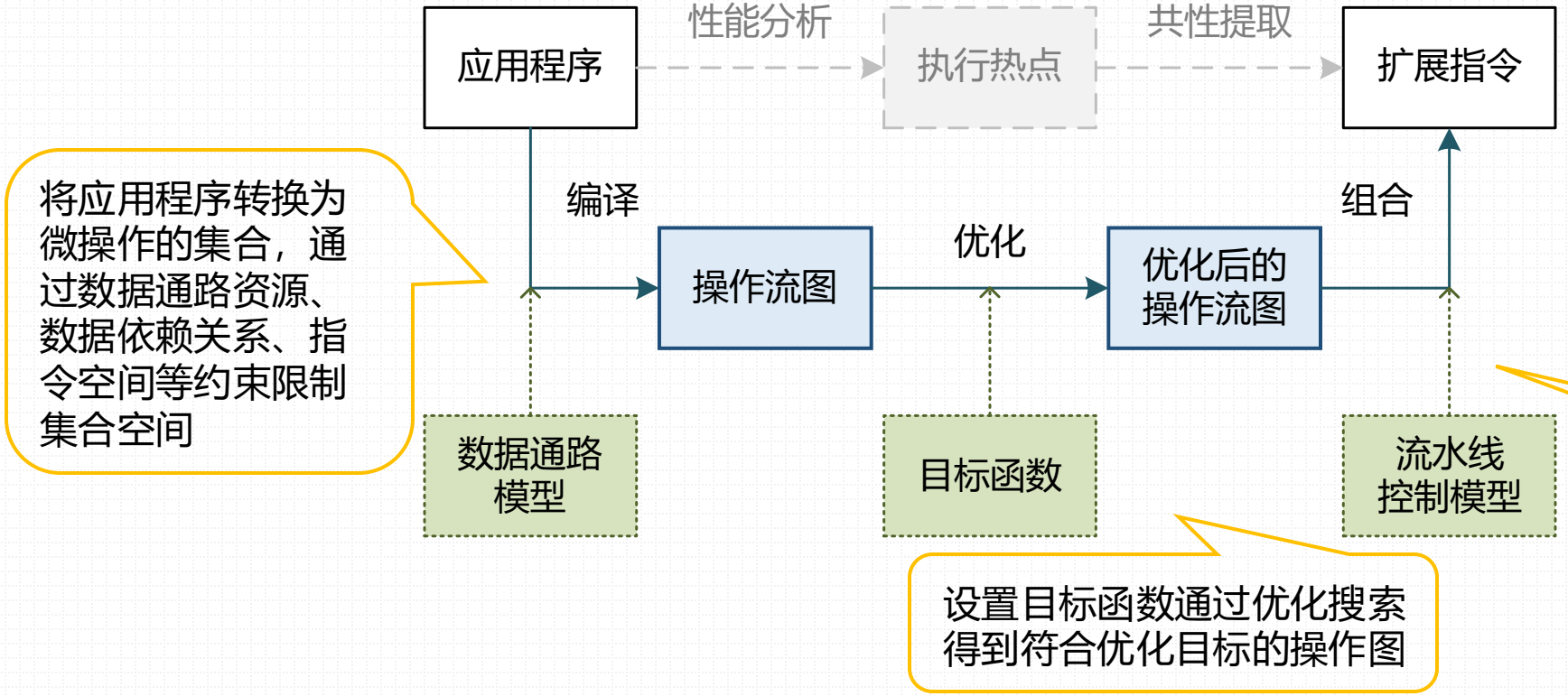
目录

目录

指令集设计方法

指令集设计视为一个优化问题

- 一个包含所有可行解的空间
- 一个用于评估一组解的目标函数



MIPS “1%原则”

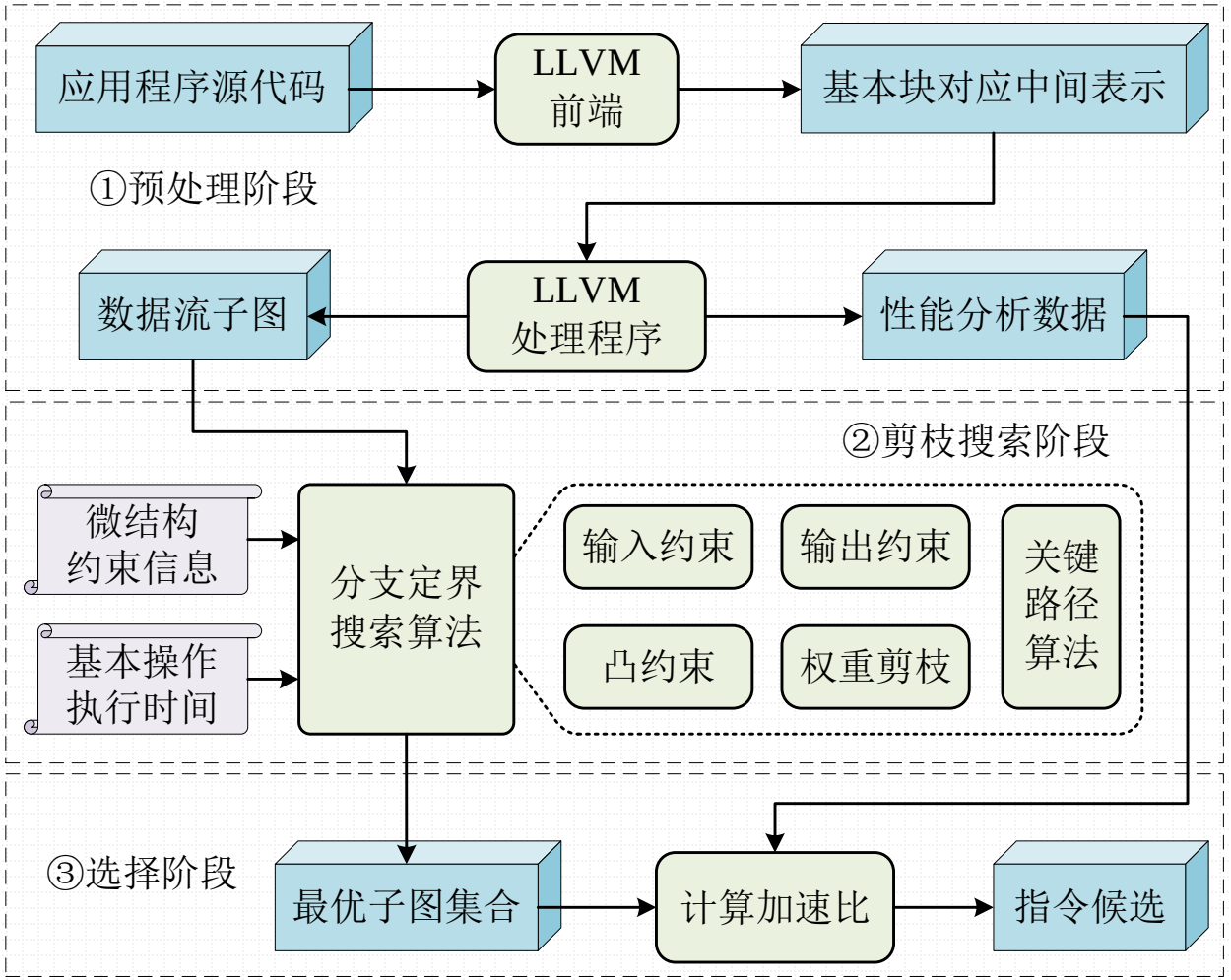
任何由于提高性能原因添加的指令都必须在一系列应用程序上提供可验证的1%性能增益

$$100\Delta C / C + \Delta I < 0$$

$$\text{最小化}\ 100\ln C + I$$

Holmer B K, Despain A M. Viewing Instruction Set Design as an Optimization Problem. MICRO, 1991.

加速指令优化识别框架



预处理阶段

- 使用LLVM生成中间表示
- 通过插桩记录程序运行数据
- 根据禁止操作集合拆分基本块

剪枝搜索阶段

- 拓扑排序二叉树搜索
- 根据单调性进行约束剪枝
- 通过软硬件执行时间计算目标函数

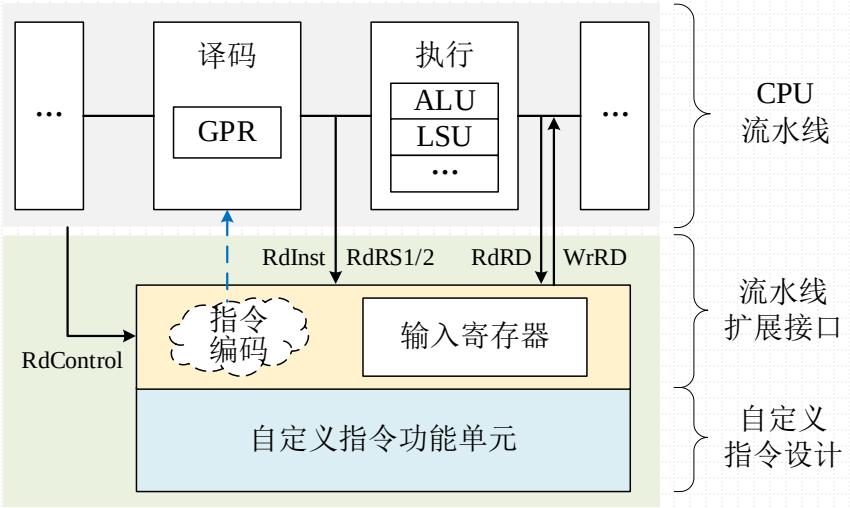
选择阶段

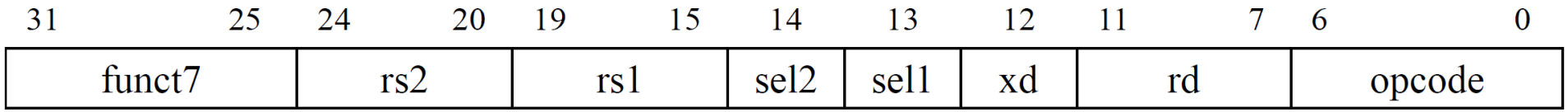
- 根据性能分析数据计算加速比

流水线扩展接口

- 通过自动化方法得到自定义指令集后，为了方便将不同自定义指令集成到RISC-V内核，定义流水线扩展接口规范
  - 在不改变原有流水线的情况下，接口为指令功能单元提供数据与控制信号

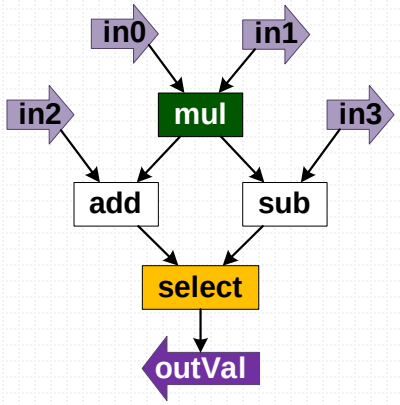
信号	位宽/bit	含义
RdRS1/RdRS2	32	从寄存器文件读取的源寄存器值
RdInst	32	正在执行的指令字
WrRD	32	写回到目的寄存器的执行结果
WrValid	1	写使能
RdRD	32	写入接口输入寄存器的指令执行结果
RdControl_X	1	内核控制信号（X = stall/flush/…）





增加输入操作数

- 为了让指令实现更复杂的功能，在接口中增加输入寄存器以增加指令的输入操作数
- 考虑输入寄存器的加载方式
  - 由于32位指令编码空间限制以及数据通路限制，每次最多从两个源寄存器中加载数据至输入寄存器
  - 设计一种加载方法，通过适当的指令排布将部分指令的执行结果同步保存到输入寄存器中



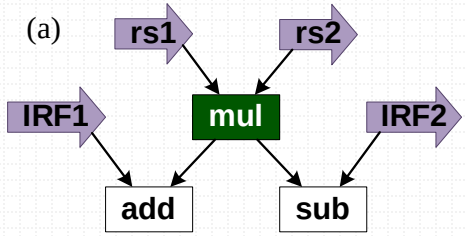
❶

```
t0 = mul i0, i1
o1 = add t0, i2
o2 = sub t0, i3
out = sel ? o1 : o2
```

❷

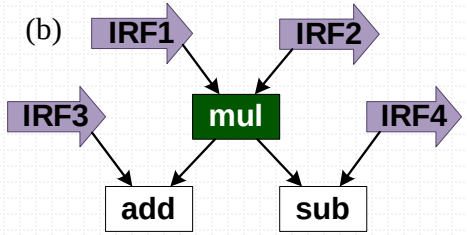
```
x10 = add x4, x5   # input i0
x11 = add x4, x6   # input i1
x12 = slli x7, 2   # input i2
x13 = sub x8, x9   # input i3
```

(a)

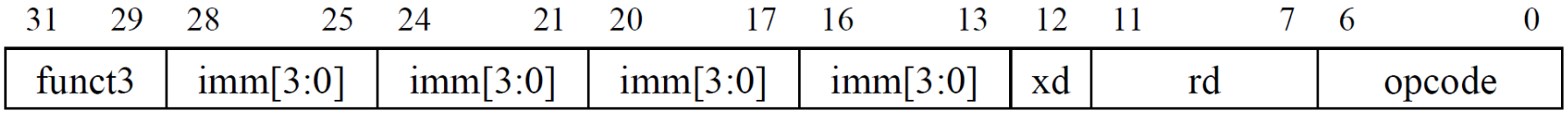



```
SET_IRF 0,0,1,2
x10 = add x4, x5   # input rs1
x11 = add x4, x6   # input rs2
x12 = slli x7, 2   # input irf1
x13 = sub x8, x9   # input irf2
x15 = EX_MX x10,x11,sel
```

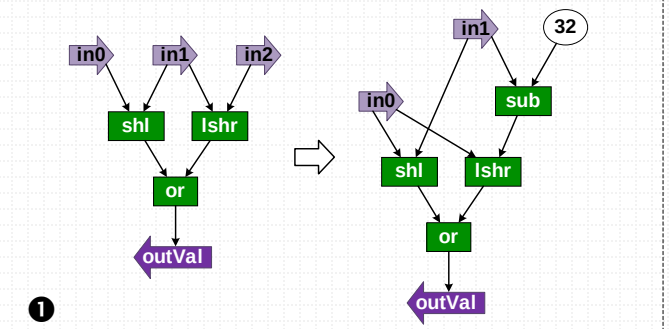
(b)

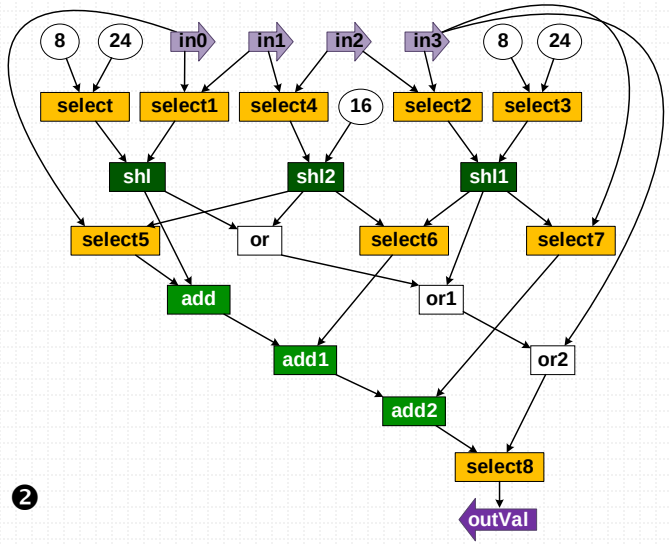


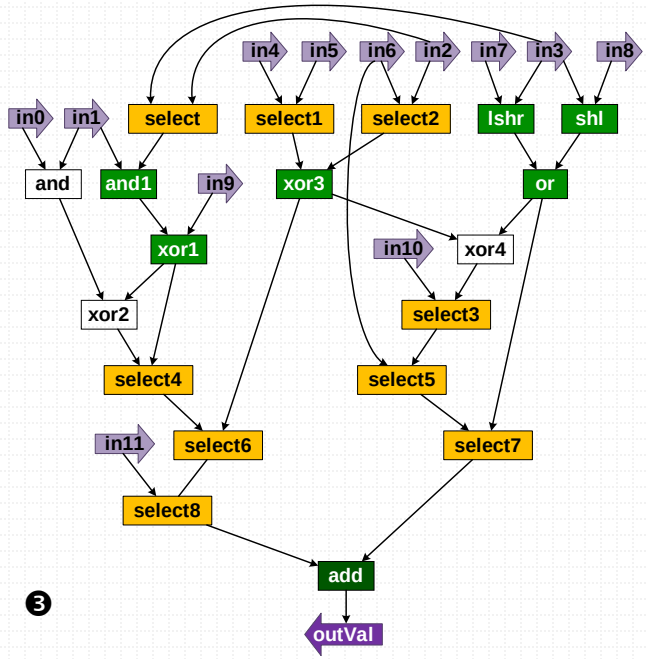

```
SET_IRF 1,2,3,4
x10 = add x4, x5   # input irf1
x11 = add x4, x6   # input irf2
x12 = slli x7, 2   # input irf3
x13 = sub x8, x9   # input irf4
x15 = EX_MX sel
```

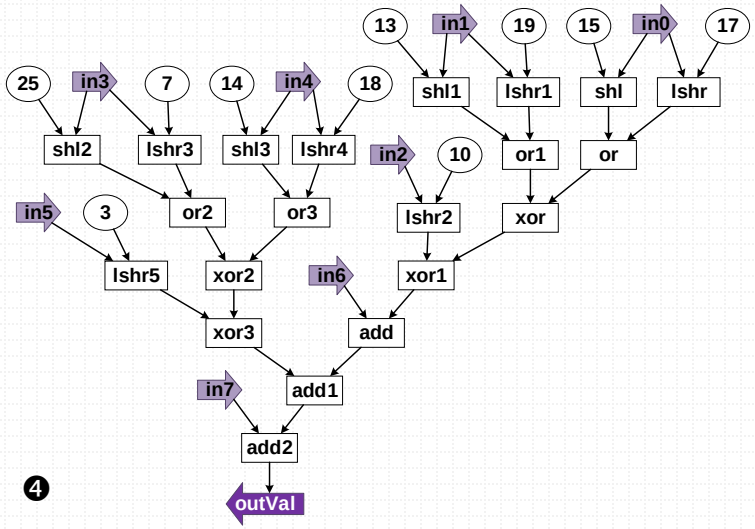


指令实现









① EX_CS: 0000000 | rs2 | rs1 |
00 | xd | rd | 1011011

SET_IRF: 000 | ir3 | ir2 | ir1 | ir0 |
0 | 00000 | 1111011

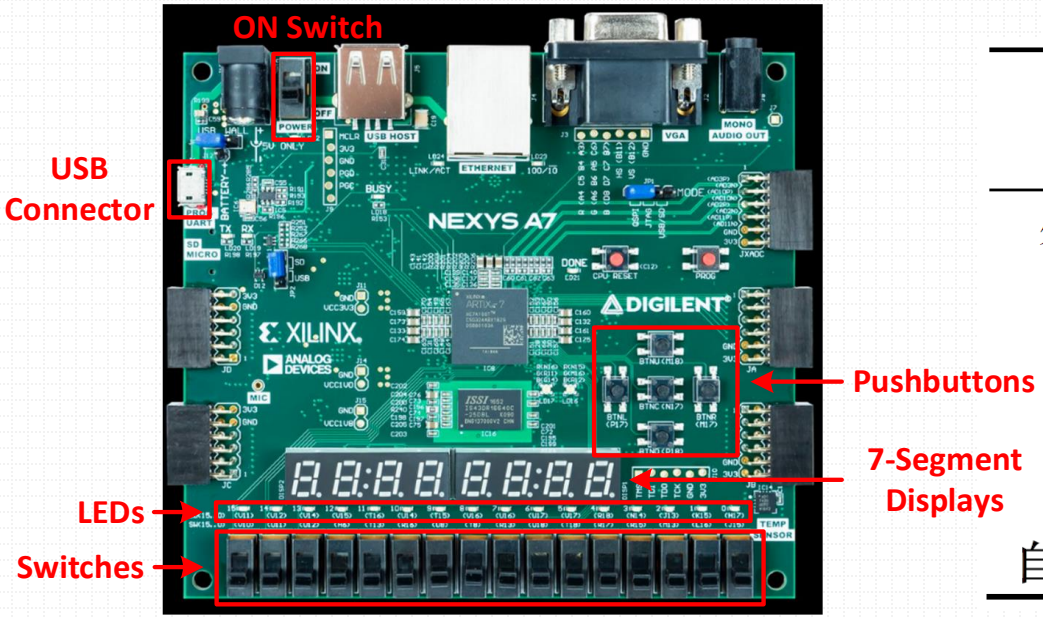
② EX_SA: 001 | sl3 | sl2 | sl1 |
sl0 | xd | rd | 1111011

③ EX_SH: 010 | sl3 | sl2 | sl1 |
sl0 | xd | rd | 1111011

④ EX_SG: 011 | 0000 | 0000 | 0000 |
0000 | xd | rd | 1111011

指令仿真与原型验证

- 根据指令编码使用**内联汇编**在应用中调用自定义指令
- 在**RISC-V**指令集仿真器**Whisper**中模拟应用执行，验证结果正确性
- 在**VeeR EH1**处理器中实现扩展接口与自定义指令，并将设计映射到**FPGA**进行验证
  - 在**Nexys A7**开发板中实现原型系统
  - 系统包含处理器子系统、通用外设、DDR存储器等模块



实现单元	查找表 LUTs		触发器 Flip-flops	
	数量	百分比	数量	百分比
处理器子系统	31 149	100	15 874	100
EX_CS	77	0.25	11	0.07
EX_SA	341	1.09	152	0.96
EX_SH	447	1.44	144	0.91
EX_SG	47	0.15	129	0.81
自定义指令开销	**912**	**2.93**	**436**	**2.75**

FPGA实现结果

测试设置

- 将待测应用数据存放在与**EH1**内核紧密耦合的数据存储器中，开启**Gshare**分支预测，减少因数据存储访问或分支跳转造成的性能损失
- 通过内核硬件计数器统计处理器事务，得到执行指令数与周期数
- 将执行结果与硬件计数器值通过**UART**打印在主机窗口中，判断程序是否正确执行并得到性能数据

	执行指令数	执行周期数	IPC	加速比
MD5 原始程序	32 376 116	33 606 636	0.963	1.00
+①	30 375 220	31 630 756	0.960	1.06
+①②	**29 374 772**	**30 955 093**	**0.949**	**1.09**
SHA1 原始程序	59 607 973	63 225 328	0.943	1.00
+①	53 855 397	57 843 428	0.931	1.09
+①②	52 854 949	55 536 857	0.952	1.14
+①②③	**46 602 149**	**49 159 023**	**0.948**	**1.29**
SHA256 原始程序	94 014 314	95 975 332	0.980	1.00
+①	84 009 834	85 170 317	0.986	1.13
+①②	83 009 386	85 013 359	0.976	1.13
+①②③	80 008 042	81 772 797	0.978	1.17
+①②③④	**65 751 658**	**66 681 981**	**0.986**	**1.44**

RISC-V指令集

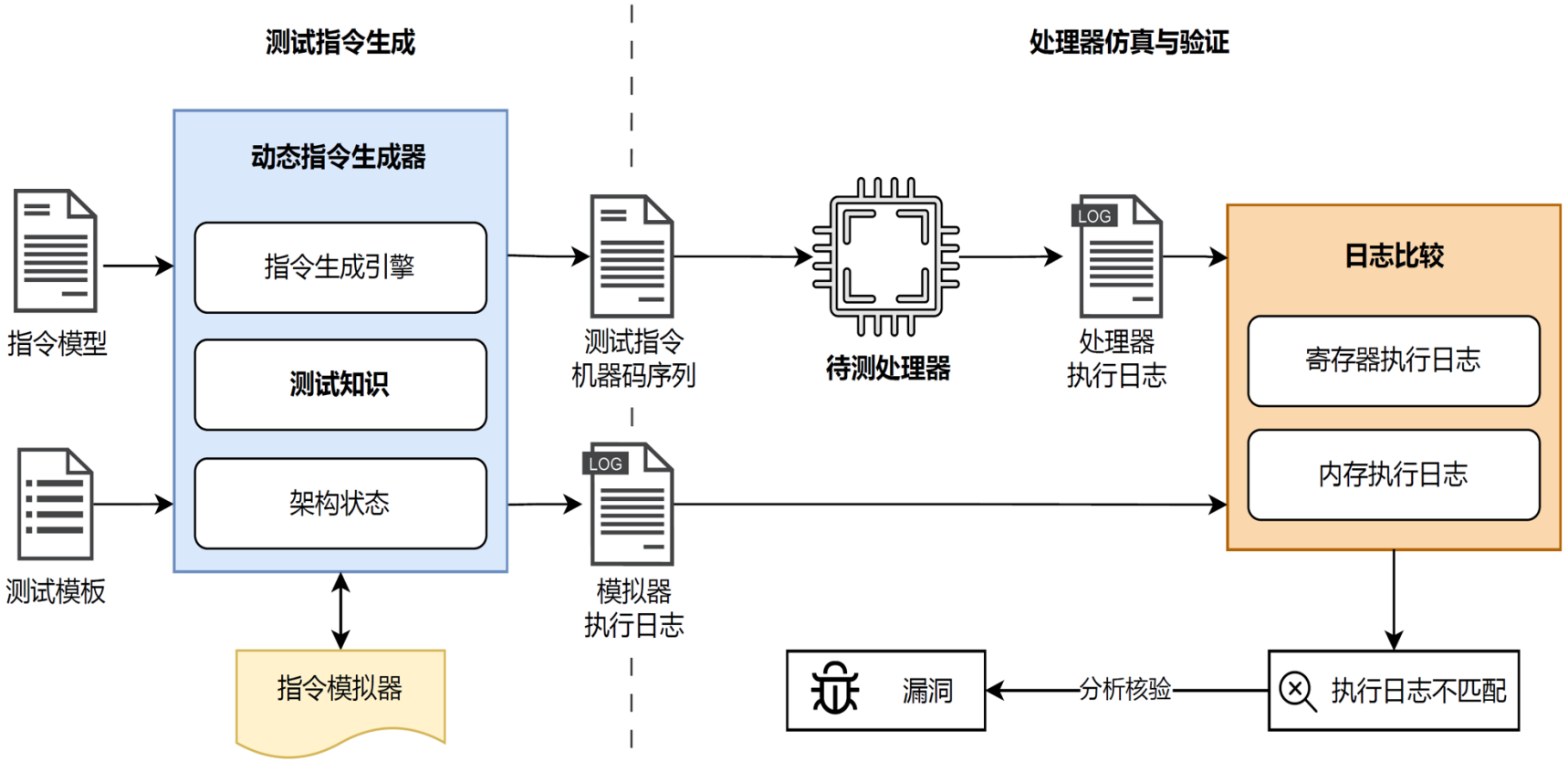
- 基础指令集**RV32I**
- **模块化的指令扩展**
- 预留了**4个定制操作码**空间

inst[4:2] / inst[6:5]	000	001	010	011	100	101	110	111 (>32b)
00	LOAD	LOAD-FP	custom-0	MISC-MEM	OP-IMM	AUIPC	OP-IMM-32	48b
01	STORE	STORE-FP	custom-1	AMO	OP	LUI	OP-32	64b
10	MADD	MSUB	NMSUB	NMADD	OP-FP	reserved	custom-2/rv128	48b
11	BRANCH	JALR	reserved	JAL	SYSTEM	reserved	custom-3/rv128	≥80b

目录

动态生成：通过测试知识利用处理器状态引导指令动态生成

- 测试指令生成
  - 输入
    - 指令模型
    - 测试模板
  - 输出
    - 指令机器码序列
    - 模拟器执行日志
- 处理器仿真与验证
  - 指令模拟
  - 执行日志比较
  - 分析不匹配信息



动态生成：通过测试知识利用处理器状态引导指令动态生成

- 寄存器的值和访问历史
- 内存的访问地址历史
- 已生成指令地址
- 程序计数器的值

- 测试知识：
  - 分支感知
  - 访存检测
  - 指令内/间测试知识

…

lw x2, 4(x1)	000000000100	00001	010	00010	0000011
lh x3, 4(x1)	000000000100	00001	001	00011	0000011

…

指令生成质量评估

实验方法学

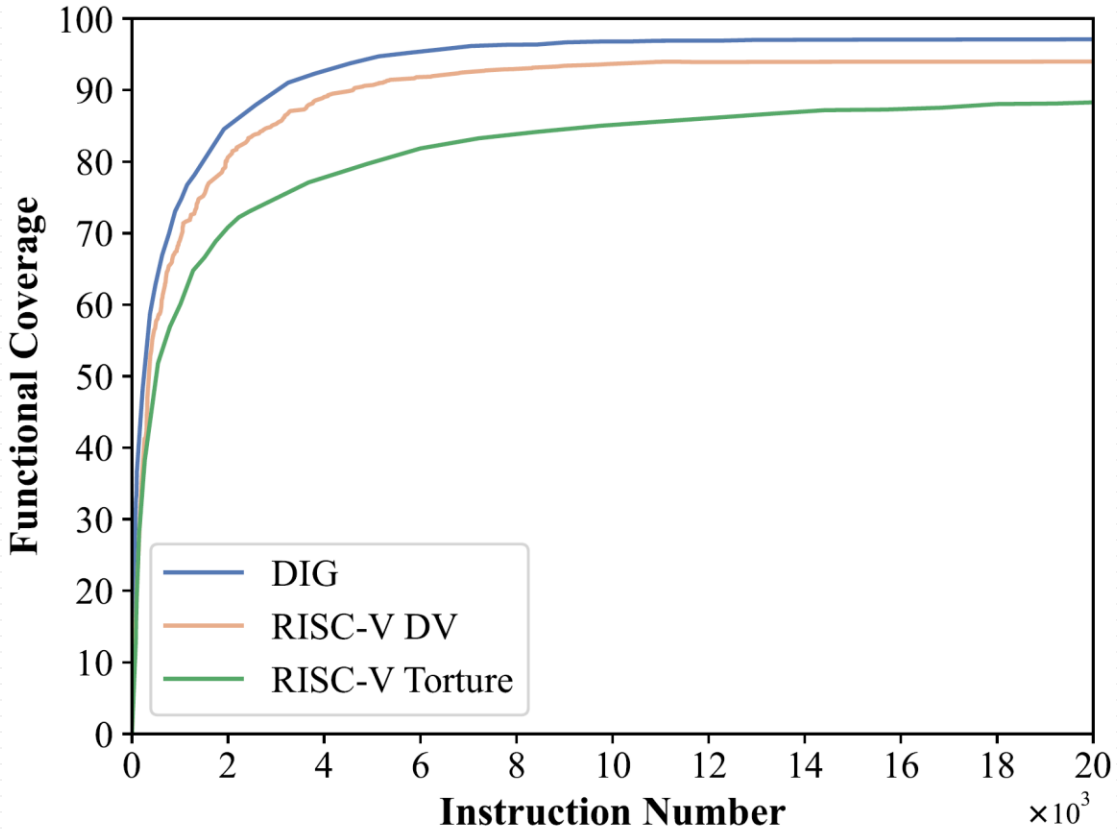
- 验证处理器：West Digital VeeR EH2，指令集：RV32I， 参考模型：QEMU
- 对比工作：RISC-V DV，RISC-V Torture

生成质量度量：功能覆盖率

- 指令操作数、冒险、对齐/未对齐加载与存储
- 正/负立即值、前向/后向分支

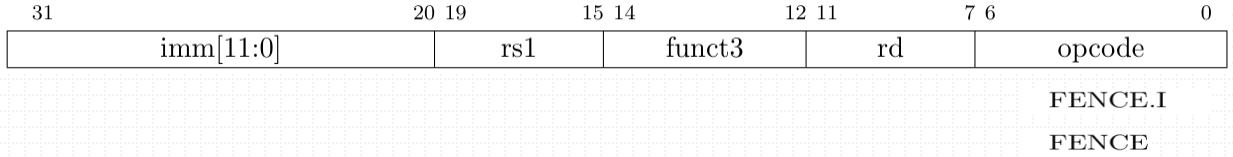
评估结果

- RISC-V DV
  - 12 000 指令，达到 92%
- RISC-V Torture
  - 18 000 指令，达到 88%
- 动态方法
  - 所需指令分别减少 62.50% 和 86.11%
  - 更高的功能覆盖率



解码错误

- **RISC-V** ***fence***、***fence.i*** 指令规范
  - 为了向前兼容，执行时必须忽略的 **rd** 字段和 **rs1** 字段
  - 标准软件应将 **rd**、**rs1**字段置零
- 非标准 ***fence*** 被视为非法指令
  - 当执行带有非零 **rd** 字段或 **rs1** 字段的指令，**EH2** 会抛出非法指令异常
  - 只有字段为零时，**EH2** 才会将该指令视为合法指令
- 非标准 ***fence.i*** 被视为非法指令
  - 与 ***fence*** 指令类似



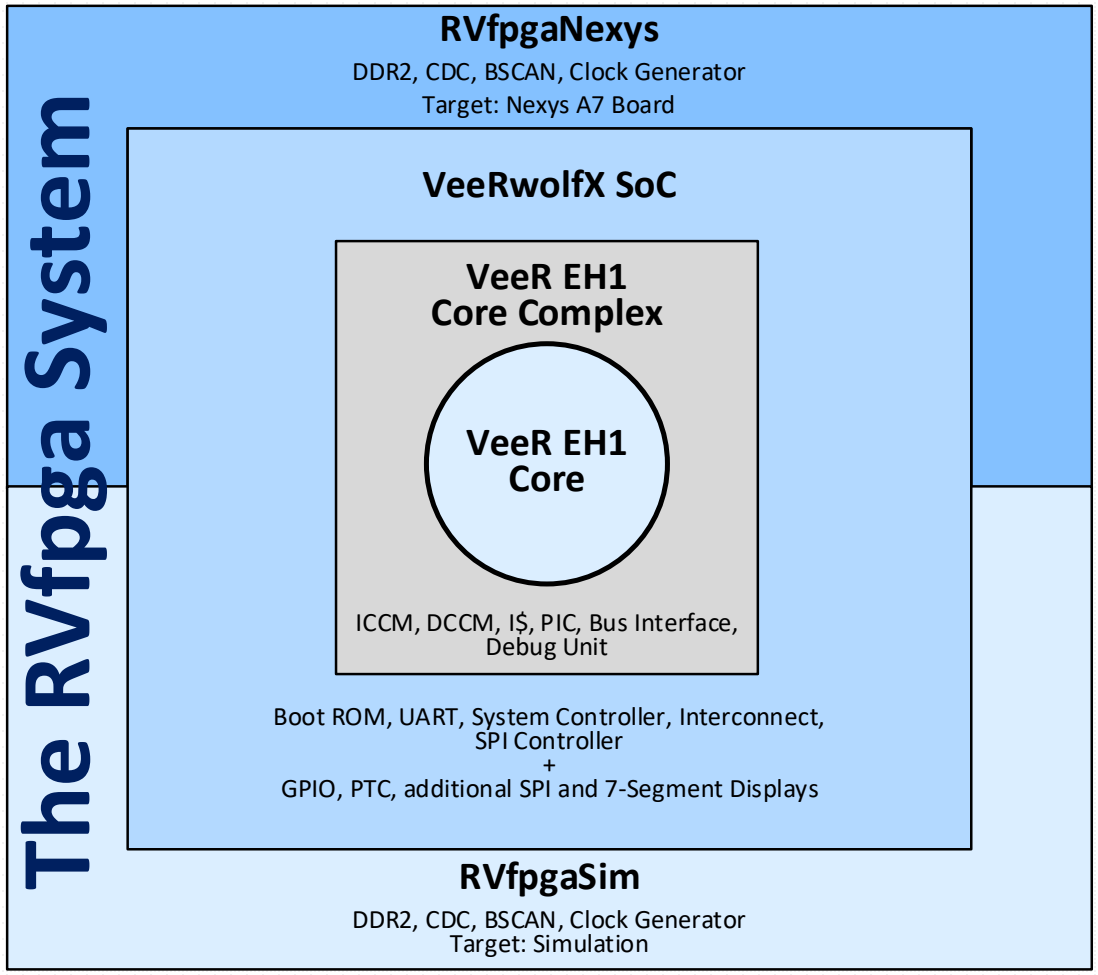
日志缺陷

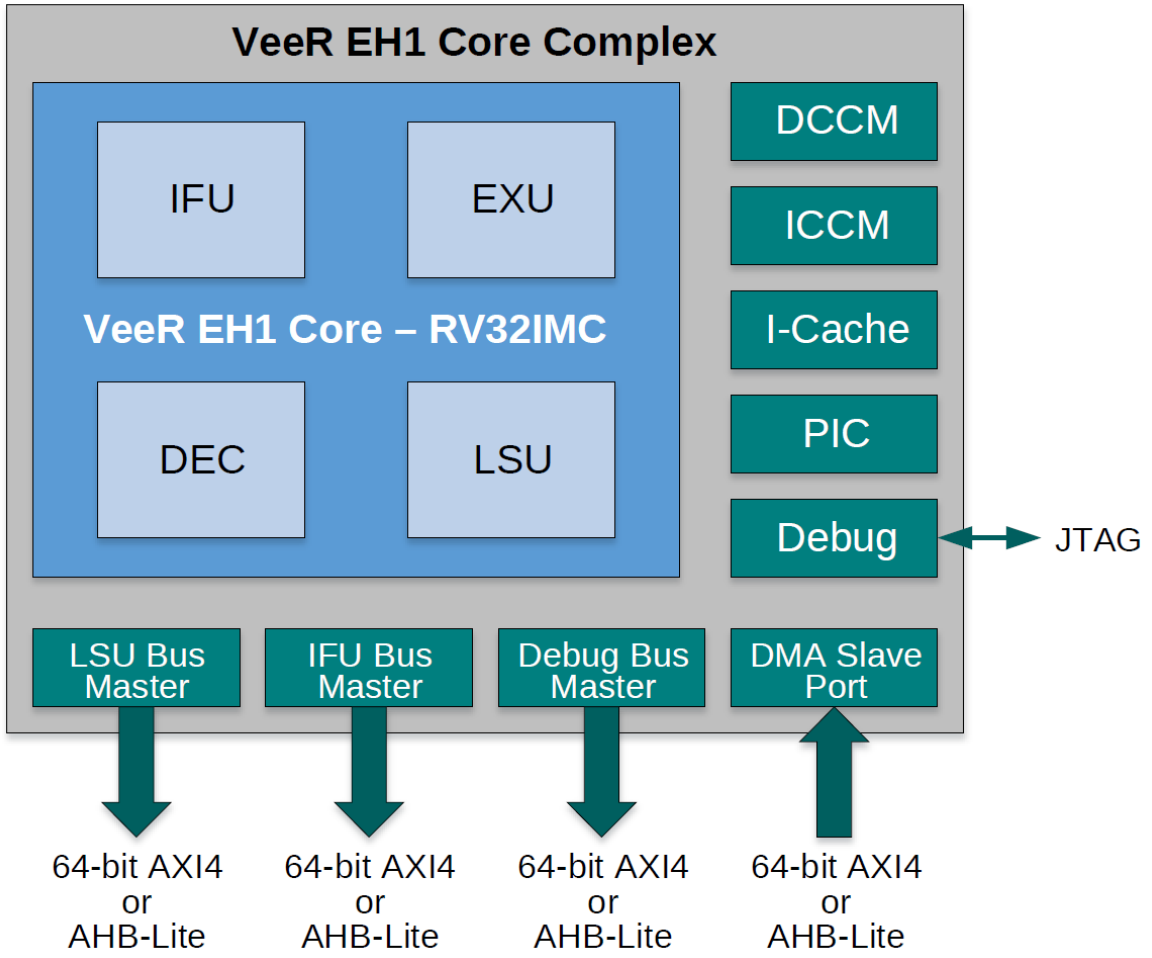
- **RISC-V** 指令规范
  - **x0**寄存器始终硬链接为0
- 目的寄存器为**x0**的加载指令执行日志中**x0**未硬链接为0
  - 寄存器文件中**x0**始终为0，不影响执行

目录

浙江大学 ZHEJIANG UNIVERSITY

RVfpga系统和内核





基准测试与编译优化

- 对处理器进行基准测试，运行程序集测定处理器性能
- **RVfpga**引入的基准：**CoreMark**和**Dhrystone**
  - 使用Chips Alliance提供的源代码对它们进行了修改，使其能够适用于**RVfpga**系统
- 对于任何基准，**硬件计数器**都将测量各种处理器事件
  - 修改基准以便使用**RISC-V**硬件计数器
- 添加对使用数据/指令紧耦合存储(DCCM/ICCM)**及编译器优化**的支持

采用量化指标用于衡量存储子系统性能

- **CoreMark指标**（多次迭代运行CoreMark得到）
  - CoreMark分数（**CM**）：每秒钟完成的迭代次数（即，迭代数/秒）
  - **CM/MHz**：CM除以单位为MHz的时钟频率（也称为Iterat/Sec/MHz，即迭代数/秒/MHz）
- **指令数**：运行过程中处理器执行的指令数目
- **周期数**：CoreMark运行消耗的时钟周期数
- **数据总线事务**：数据总线上数据传输事务的数量
- **指令总线事务**：指令总线上数据传输事务的数量

量化指标	编译器 调试外部存储器	编译器 调试DCCM	编译器 优化DCCM
CM/MHz	0.47	1.88	3.47
指令数	50万	50万	30.9万
周期数	200万	50万	28.8万
IPC	0.25	1	1
数据总线事务	133 000 (外部存储器)	0 (由于DCCM)	0 (由于DCCM)
指令总线事务	392 (由于I$)	392 (由于I$)	392 (由于I$)

目录

➢ 加速指令的设计

- 循序渐进引导同学熟悉RISC-V指令
- 结合具体**哈希算法**应用，指导设计加速指令
- 哈希算法执行指令数**量化**衡量指令加速效果

➢ 指令实现与验证

- 修改**RVfpga**项目代码，实现指令设计
- 使用**指令生成工具**验证指令设计的一致性

➢ 存储与编译优化

- **RVfpga**数据/指令紧耦合存储器DCCM/ICCM
- 分析使用**DCCM/ICCM及编译优化**前后系统性能

“计算机组成与设计”课程实践
基于RVfpga的量化研究

特别感谢**Imagination Tech. Robert Owen**, 许可, 刘敬阳, 田苗等人3年来的大力支持!
特别感谢**浙江大学信息电子工程学院钟婷婷**, 吴叶飞, 杨建义, 史治国等老师的协助!