NutShell-本科生设计的可运行Linux的RISC-V芯片

中国科学院大学
王华强 张紫飞 金越 张林隽 王凯帆
2020.7.18

中国科学院大学
University of Chinese Academy of Sciences

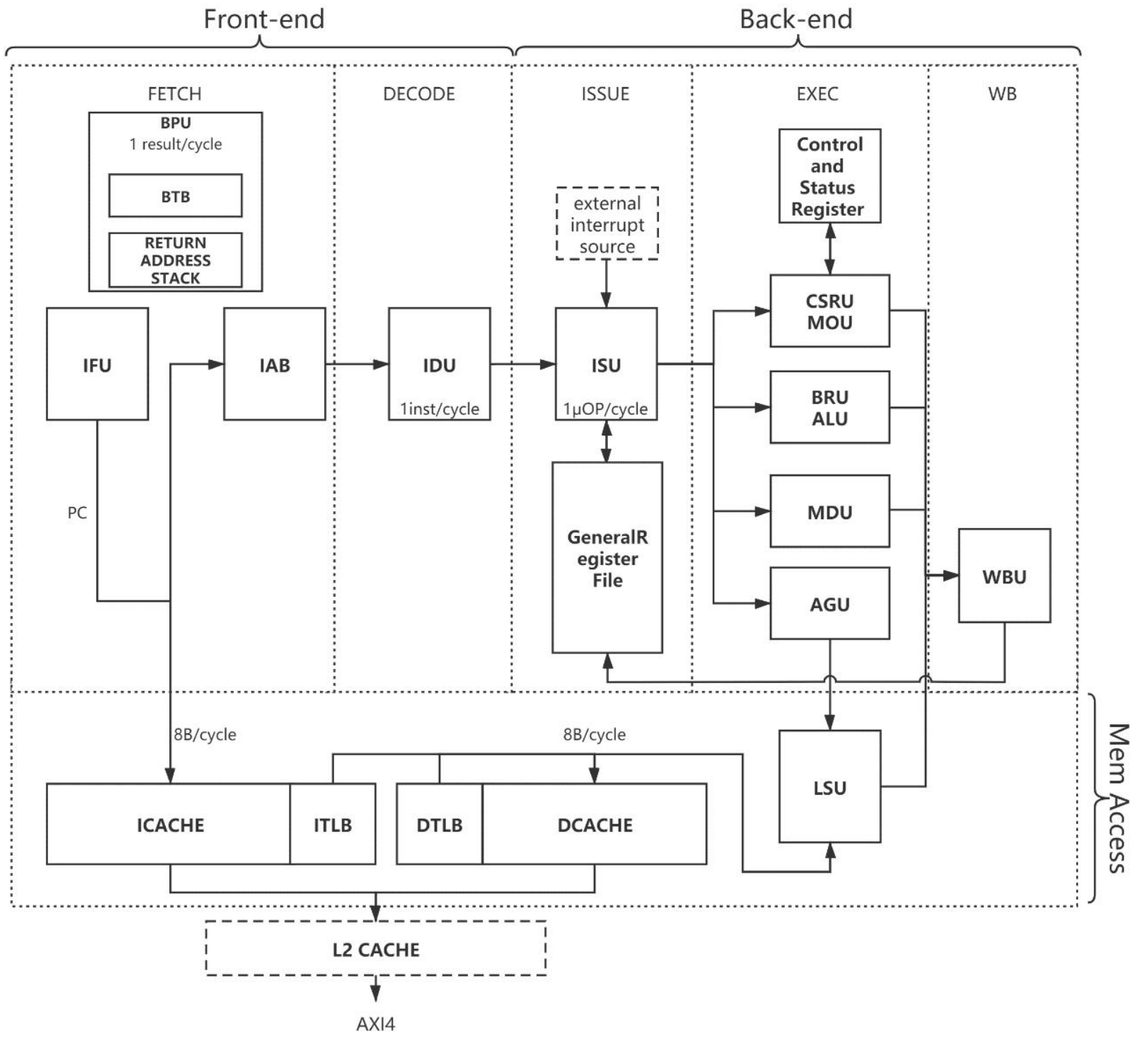
目录

- NutShell简介
- 经验分享
  - 使用Chisel进行硬件开发
  - 差分测试框架

NutShell简介

顺序RISC-V核: NutCore

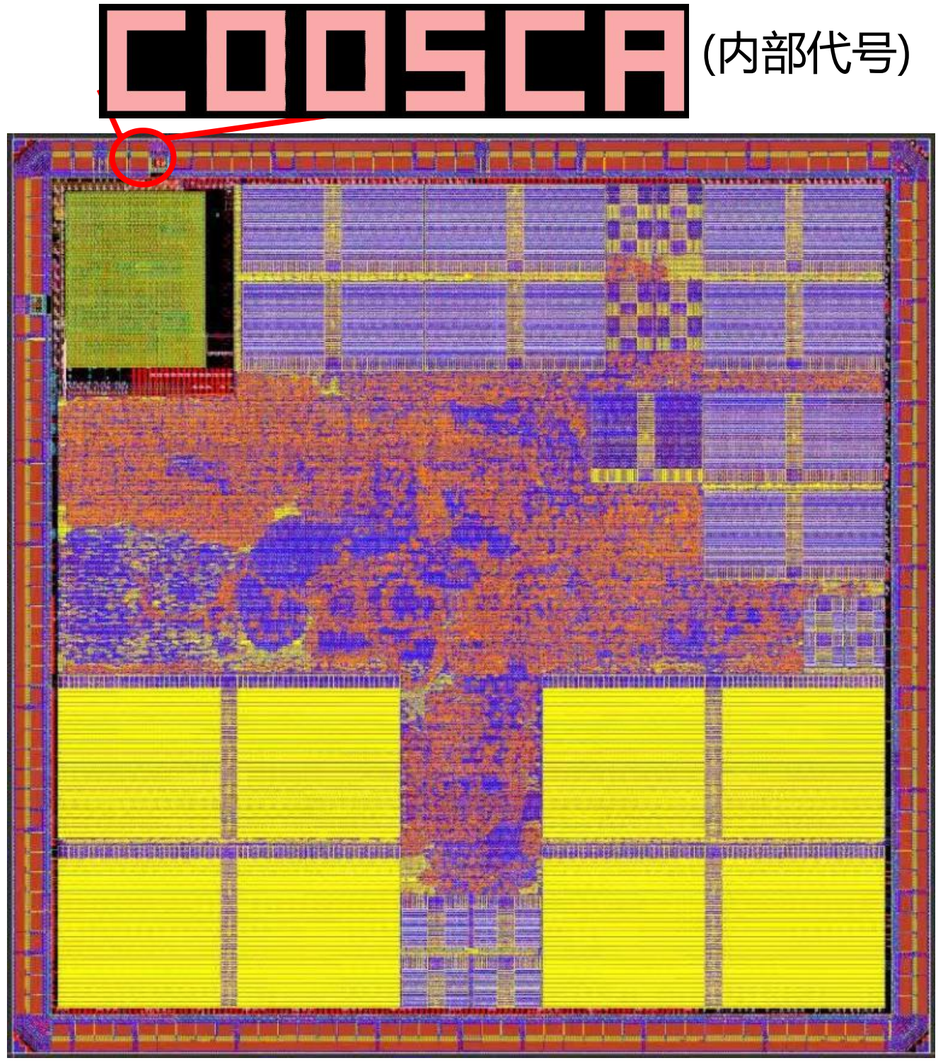
- **单发射顺序核**
- RV64IMAC指令集，支持M/S/U特权级
- 两位饱和计数器分支预测
- 512项BTB，16项RAS
- **支持Sv39分页机制, 支持硬件TLB填充**
- 32K指令/数据L1 cache
- 支持L1 指令/数据cache读一致性
- 128K L2 cache，支持next line预取
- **使用Chisel语言开发**



SoC芯片: NutShell (果壳)

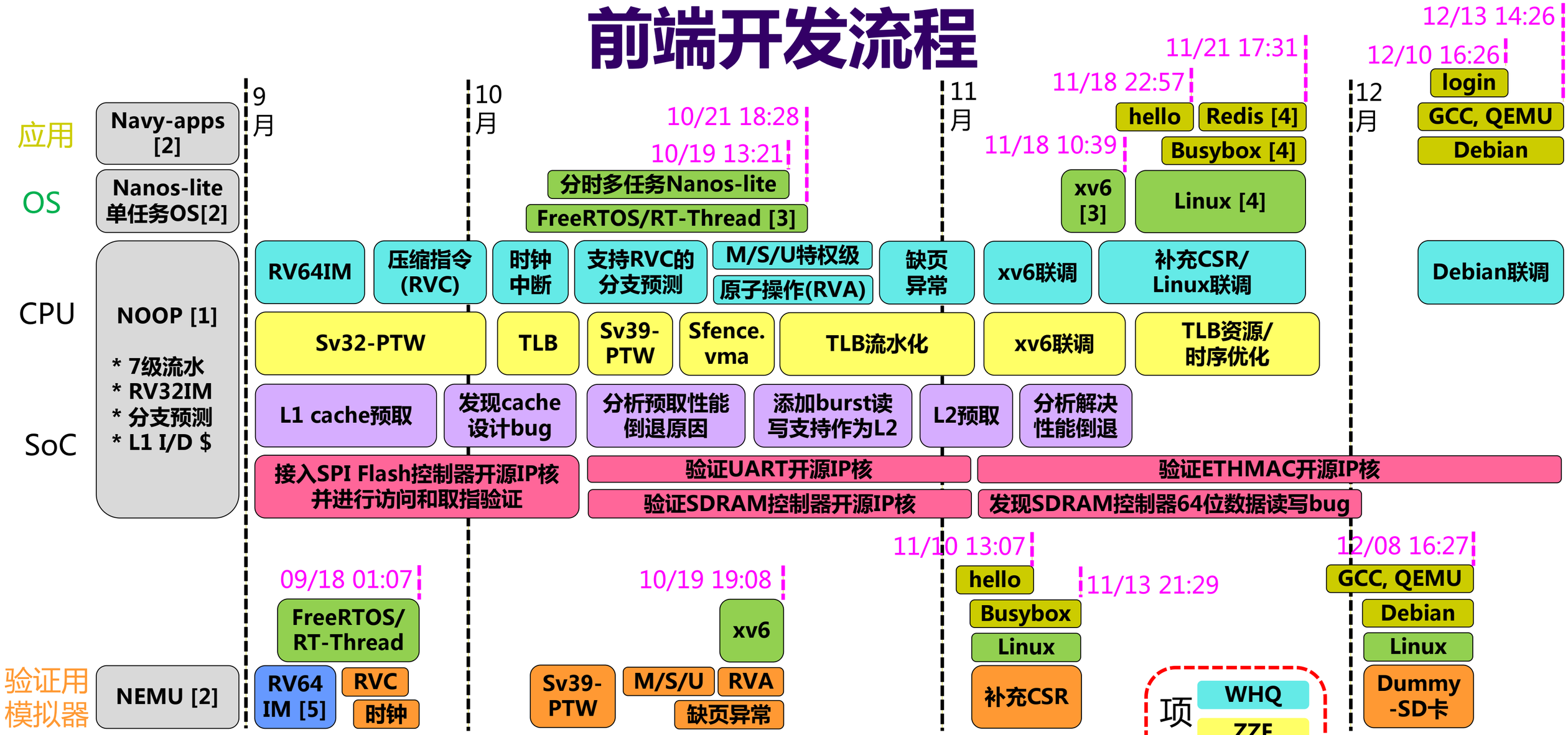
- **使用Chisel语言开发[1]**
- 支持SDRAM 、SPI flash、UART等外设
- 支持启动**Linux 4.18.0内核**
- 支持运行Busybox套件
- 在仿真/FPGA环境下支持启动**Debian 11**

- SMIC 110nm工艺
- 10mm^2
- **200mw@350MHz** Typical
- TQFP100封装



[1] 外设部分仍使用Verilog进行开发

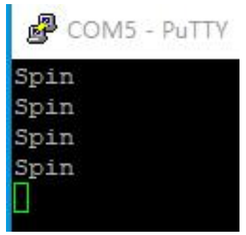
前端开发流程



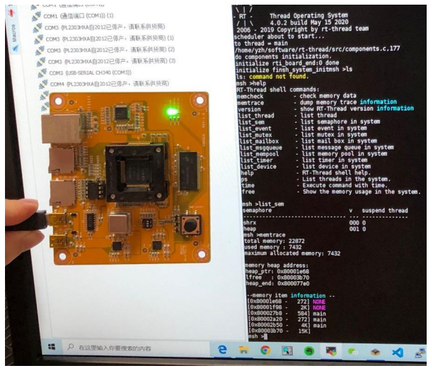
[1] 芯片开发基于南京大学的教学处理器NOOP, 以运行Linux + Debian的完整RISC-V芯片为目标进行改进
[2] 使用了南京大学计算机系统教学生态Project-N中的若干子项目协助芯片验证, https://github.com/NJU-ProjectN
[3] FreeRTOS/RT-Thread/xv6来源于开源社区, 并进行了少量移植工作
[4] Linux, Busybox和Redis使用了计算所Labeled RISC-V项目中提供的开源分支, https://github.com/LvNA-system
[5] WKF同学于9月~12月在国外访学, 访学结束后负责项目的可配置化以及在科研项目中的应用

NutShell测试流程和功能展示

封装后的芯片



串口首次输出



成功运行
RT-Thread

本科
毕业
答辩

在本科毕业远程答辩中，代表一生一芯团队向评委展示芯片运行的过程

5月

4/25 12:32
4/23 14:56
封装
完成
芯片装入测试板，
开始串口测试

5/15 22:48
5/15 10:00
串口测试
RT-Thread
修复电源问题
生产第二版测试板
SDRAM
测试

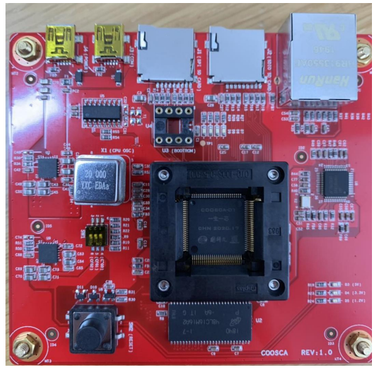
5/19 16:43
更换正确型号的
SDRAM颗粒
200MHz
测试
SDRAM
测试

5/28 19:54
5/27 21:00
hello
Busybox
xv6
修改版
Linux

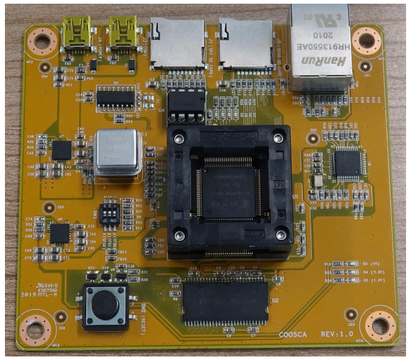
6/2 13:30
6月

6/7 22:54
Busybox
原版
Linux
200MHz
测试

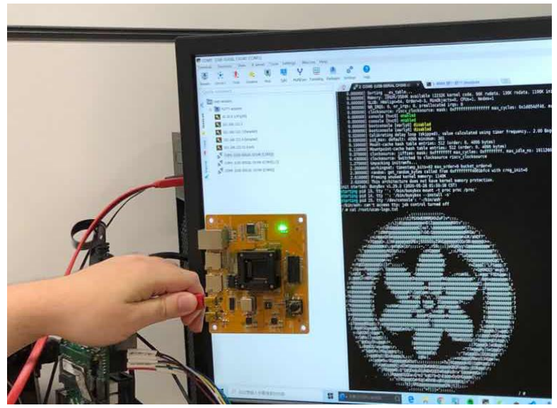
由于SD控制器的缺陷，无法在芯片上启动Debian
SD卡
测试



第一版
测试板



第二版
测试板



成功运行Linux
并展示中科院logo

分工
YZH
CY
LT

学生成员因疫情无法参与芯片测试流程

NutShell仿真结果

```
bbl loader
                           vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
                            vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
rrrrrrrrrrrrr       vvvvvvvvvvvvvvvvvvvvvvvvvvv
rrrrrrrrrrrrrrrr      vvvvvvvvvvvvvvvvvvvvvvvvv
rrrrrrrrrrrrrrrrrr    vvvvvvvvvvvvvvvvvvvvvvvvv
rrrrrrrrrrrrrrrrrr    vvvvvvvvvvvvvvvvvvvvvvvvv
rrrrrrrrrrrrrrrrrr    vvvvvvvvvvvvvvvvvvvvvvvvv
rrrrrrrrrrrrrrrr      vvvvvvvvvvvvvvvvvvvvvvv
rrrrrrrrrrrrr       vvvvvvvvvvvvvvvvvvvvv
rr               vvvvvvvvvvvvvvvvvvvvv
rr           vvvvvvvvvvvvvvvvvvvvvvvvv      rr
rrrr      vvvvvvvvvvvvvvvvvvvvvvvvv      rrrr
rrrrrr      vvvvvvvvvvvvvvvvvvvvv      rrrrrr
rrrrrrrr      vvvvvvvvvvvvvvvvv      rrrrrrrr
rrrrrrrrrr      vvvvvvvvvvvvv      rrrrrrrrrr
rrrrrrrrrrrr      vvvvvvvvv      rrrrrrrrrrrr
rrrrrrrrrrrrrr      vvvvvv      rrrrrrrrrrrrrr
rrrrrrrrrrrrrrrr      vv      rrrrrrrrrrrrrrrr
rrrrrrrrrrrrrrrrrr          rrrrrrrrrrrrrrrrrr
rrrrrrrrrrrrrrrrrrrr      rrrrrrrrrrrrrrrrrrrr
rrrrrrrrrrrrrrrrrrrrrr  rrrrrrrrrrrrrrrrrrrrrr

       INSTRUCTION SETS WANT TO BE FREE
[    0.000000] OF: fdt: Ignoring memory range 0x80000000 - 0x80200000
[    0.000000] Linux version 4.18.0-ga2ee7cb2-dirty (yzh@i7-9700k) (gcc version 9.2.0 (GCC)) #25 Tue Dec 10 12:49:30
 CST 2019
[    0.000000] bootconsole [early0] enabled
[    0.000000] Zone ranges:
[    0.000000]   DMA32    empty
[    0.000000]   Normal   [mem 0x0000000080200000-0x000000008fffffff]
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x0000000080200000-0x000000008fffffff]
[    0.000000] Initmem setup node 0 [mem 0x0000000080200000-0x000000008fffffff]
[    0.000000] software IO TLB [mem 0x8bc7d000-0x8fc7d000] (64MB) mapped at [(____ptrval____)-(____ptrval____)]
[    0.000000] elf_hwcap is 0x112d
[    0.000000] Built 1 zonelists, mobility grouping on.  Total pages: 64135
[    0.000000] Kernel command line: root=/dev/mmcblk0 rootfstype=ext4 ro rootwait earlycon
[    0.000000] Dentry cache hash table entries: 32768 (order: 6, 262144 bytes)
[    0.000000] Inode-cache hash table entries: 16384 (order: 5, 131072 bytes)
[    0.000000] Sorting __ex_table...
[    0.000000] Memory: 184920K/260096K available (3138K kernel code, 219K rwdata, 946K rodata, 116K init, 774K bss,
75176K reserved, 0K cma-reserved)
[    0.000000] SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
[    0.000000] NR_IRQS: 0, nr_irqs: 0, preallocated irqs: 0
[    0.000000] clocksource: riscv_clocksource: mask: 0xffffffffffffffff max_cycles: 0x1d854df40, max_idle_ns: 352636
1616960 ns
[    0.000000] Console: colour dummy device 80x25
[    0.000000] console [hvc0] enabled
[    0.000000] console [hvc0] enabled
[    0.000000] bootconsole [early0] disabled
[    0.000000] bootconsole [early0] disabled
[    0.000000] Calibrating delay loop (skipped), value calculated using timer frequency.. 2.00 BogoMIPS (lpj=10000)
[    0.000000] pid_max: default: 32768 minimum: 301
[    0.000000] Mount-cache hash table entries: 512 (order: 0, 4096 bytes)
[    0.000000] Mountpoint-cache hash table entries: 512 (order: 0, 4096 bytes)
[    0.000000] devtmpfs: initialized
[    0.000000] random: get_random_u32 called from bucket_table_alloc+0x6c/0x1a6 with crng_init=0
[    0.000000] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 19112604462750000 ns
[    0.000000] futex hash table entries: 256 (order: 0, 6144 bytes)
[    0.000000] NET: Registered protocol family 16
[    0.020000] SCSI subsystem initialized
[    0.020000] clocksource: Switched to clocksource riscv_clocksource
[    0.030000] NET: Registered protocol family 2
[    0.030000] tcp_listen_portaddr_hash hash table entries: 256 (order: 0, 4096 bytes)
[    0.030000] TCP established hash table entries: 2048 (order: 2, 16384 bytes)
[    0.030000] TCP bind hash table entries: 2048 (order: 2, 16384 bytes)
[    0.030000] TCP: Hash tables configured (established 2048 bind 2048)
[    0.030000] UDP hash table entries: 256 (order: 1, 8192 bytes)
[    0.030000] UDP-Lite hash table entries: 256 (order: 1, 8192 bytes)
[    0.030000] NET: Registered protocol family 1
[    0.030000] RPC: Registered named UNIX socket transport module.
[    0.030000] RPC: Registered udp transport module.
```

启动bootloader
加载Linux内核

启动Linux内核

```
Debian GNU/Linux bullseye/sid riscv64-COOSCA hvc0

riscv64-COOSCA login: root
Linux riscv64-COOSCA 4.18.0-ga2ee7cb2-dirty #44 Wed Dec 25 11:58:01 CST 2019 riscv64

The programs included with the Debian GNU/Linux sys
the exact distribution terms for each program are d
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

============ Now running preset commands =============

root@riscv64-COOSCA:/root# uname -a
Linux riscv64-COOSCA 4.18.0-ga2ee7cb2-dirty #44 Wed Dec 25 11:58:01 CST 2019 riscv64 GNU/Linux

root@riscv64-COOSCA:/root# cat /etc/issue
Debian GNU/Linux bullseye/sid \n \l


root@riscv64-COOSCA:/root# cat /proc/cpuinfo
hart    : 0
isa     : rv64imac
mmu     : sv39
uarch   : UCAS,COOSCA1.0


root@riscv64-COOSCA:/root# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       3.9G  952M  2.8G  26% /
devtmpfs         91M     0   91M   0% /dev
tmpfs            91M     0   91M   0% /dev/shm
tmpfs            91M  2.3M   89M   3% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs            91M     0   91M   0% /sys/fs/cgroup
ramdisk          10M     0   10M   0% /tmp

root@riscv64-COOSCA:/root# free -h
              total        used        free      shared  buff/cache   available
Mem:          180Mi        16Mi       133Mi       2.0Mi        31Mi       155Mi
Swap:            0B          0B          0B

root@riscv64-COOSCA:/root# date
Fri Nov 29 16:33:24 UTC 2019

root@riscv64-COOSCA:/root# uptime
 16:33:24 up 0 min,  1 user,  load average: 0.42, 0.14, 0.05

root@riscv64-COOSCA:/root# ls /root
hello  nemu  nemu-prog  nemutrap

root@riscv64-COOSCA:/root# ls /root/hello
hello.c

root@riscv64-COOSCA:/root# cat /root/hello/hello.c
#include <stdio.h>

int main() {
  printf("Hello riscv64-COOSCA!\n");
  return 0;
}

root@riscv64-COOSCA:/root# gcc -time /root/hello/hello.c -o /tmp/hello
# cc1 0.19 0.05
# as 0.02 0.02
# collect2 0.19 0.04

root@riscv64-COOSCA:/root# ls -lh /tmp
total 12K
-rwxr-xr-x 1 root root 8.2K Nov 29 16:33 hello

root@riscv64-COOSCA:/root# /tmp/hello
Hello riscv64-COOSCA!
```

登录Debian最新发行版bullseye

依次查看以下系统信息
内核版本
发行版信息
CPU信息
文件系统信息
内存信息
日期
启动时间

展示hello.c文件内容

使用gcc编译hello.c

运行hello程序

NutShell性能展示

由于功能缺陷，只能配备50MHz晶振来使用SDRAM

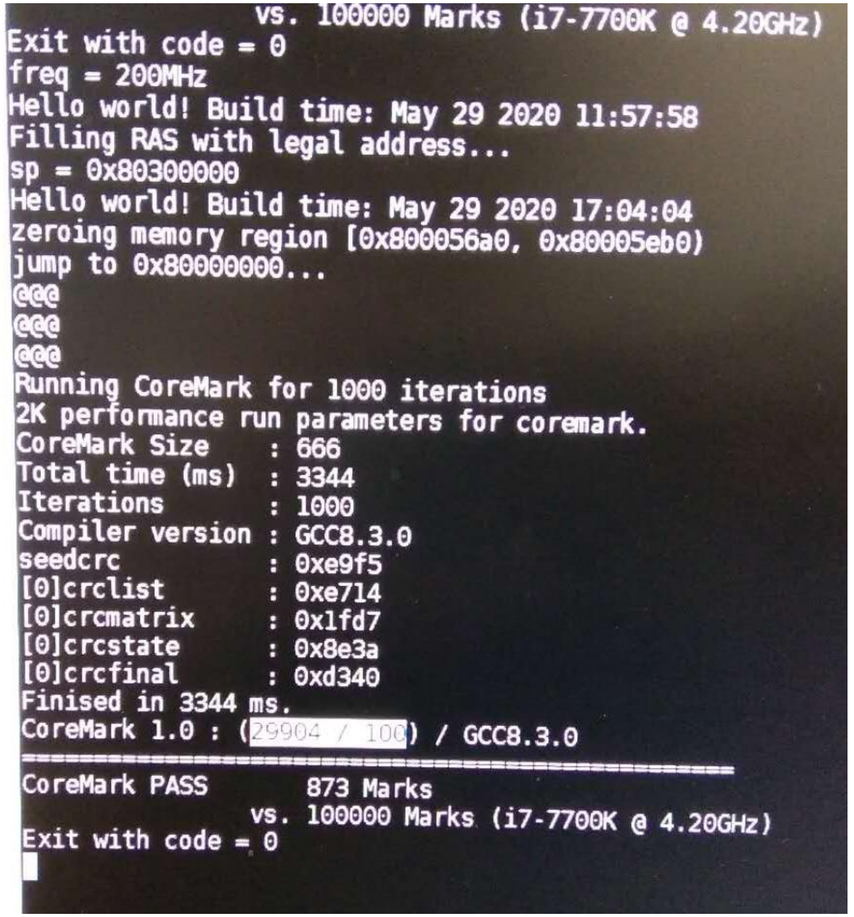
芯片工作频率

配置开关	倍率	50MHz晶振	100MHz晶振
000	1	50MHz	100MHz
001	1.5	75MHz	150MHz
010	2	100MHz	200MHz
011	2.5	125MHz	250MHz
100	2.75	137.5MHz	275MHz
101	3	150MHz	300MHz
110	3.5	175MHz	350MHz
111	4	200MHz	400MHz

使用SDRAM（如运行Linux）最高频率

无输出

不使用SDRAM（如运行RT-thread）最高频率，与后端时序报告一致



CoreMark跑分为1.49/MHz

尝试芯片设计验证的前沿技术

使用Chisel语言进行敏捷开发

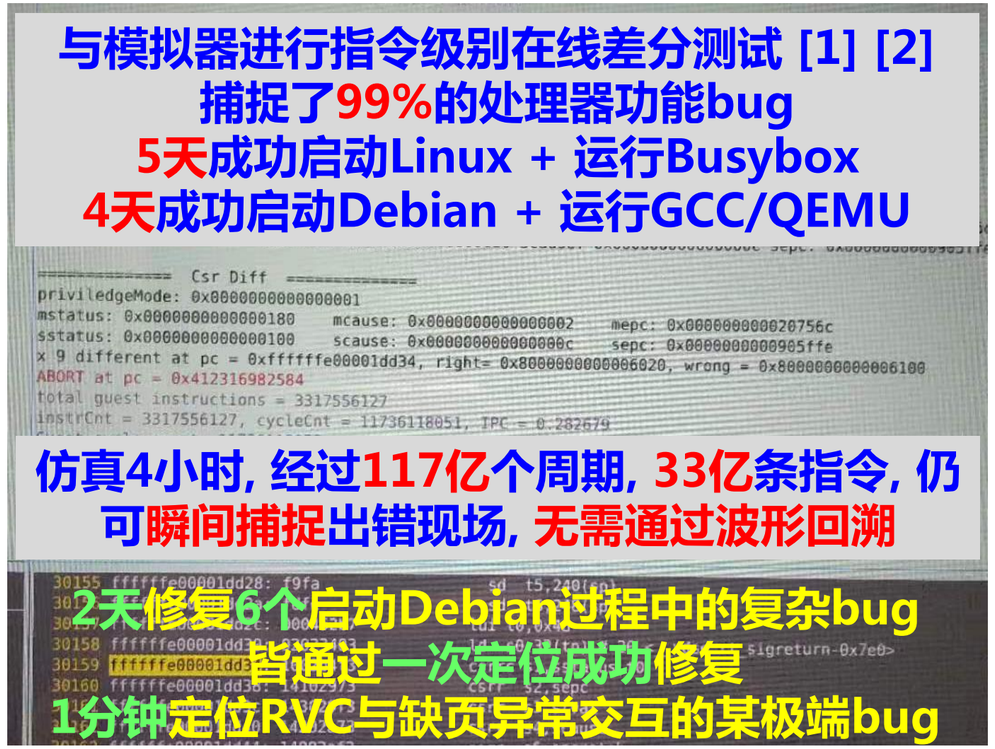
```
val ifu  = Module(new IFU)
val idu1 = Module(new IDU1)
val idu2 = Module(new IDU2)
val isu  = Module(new ISU)
val exu  = Module(new EXU)
val wbu  = Module(new WBU)

def PipelineConnect2[T <: Data](left: DecoupledIO[T], right: DecoupledIO[T],
  isFlush: Bool, entries: Int = 4, pipe: Boolean = false) = {
  right <> FlushableQueue(left, isFlush,  entries = entries, pipe = pipe)
}

PipelineConnect2(ifu.io.out, idu1.io.in, ifu.io.flushVec(0))
PipelineConnect(idu1.io.out, idu2.io.in, idu2.io.out.fire(), ifu.io.flushVec(1))
PipelineConnect(idu2.io.out, isu.io.in, isu.io.out.fire(), ifu.io.flushVec(1))
PipelineConnect(isu.io.out, exu.io.in, exu.io.out.fire(), ifu.io.flushVec(2))
PipelineConnect(exu.io.out, wbu.io.in, true.B, ifu.io.flushVec(3))
idu1.io.flush := ifu.io.flushVec(1)
idu2.io.flush := ifu.io.flushVec(1)
isu.io.flush := ifu.io.flushVec(2)
exu.io.flush := ifu.io.flushVec(3)
```

国内首个使用Chisel设计的支持Linux+Debian的开源RISC-V核

借鉴软件工程领域的差分测试技术进行处理器的仿真验证



[1] Yu, EasyDiff: An Effective and Efficient Framework for Processor Verification, CRVF 2019, https://crvf2019.github.io/pdf/14.pdf
[2] 2018年龙芯杯南京大学二队决赛答辩 报告 http://www.nscscc.org/a/wangjie/NSCSCC2018/2018/1010/46.html

使用Chisel进行硬件开发

Chisel 不是 HLS

- Chisel是硬件描述语言

```
val hitVec = VecInit(metaWay.map(
  m => m.valid && (m.tag === addr.tag) && io.in.valid
)).asUInt
```

- 可以通过Scala中的函数式编程、面向对象特性等来**方便地描述电路**
- 仍能做到对电路的**精确控制**
  - HLS描述算法, **失去了**对电路的精确控制

```
assign _T_24 = metaWay_0_tag == addr_tag; // @[Cache.scala 173:59]
assign _T_25 = metaWay_0_valid & _T_24; // @[Cache.scala 173:49]
assign _T_26 = _T_25 & io_in_valid; // @[Cache.scala 173:73]
assign _T_27 = metaWay_1_tag == addr_tag; // @[Cache.scala 173:59]
assign _T_28 = metaWay_1_valid & _T_27; // @[Cache.scala 173:49]
assign _T_29 = _T_28 & io_in_valid; // @[Cache.scala 173:73]
assign _T_30 = metaWay_2_tag == addr_tag; // @[Cache.scala 173:59]
assign _T_31 = metaWay_2_valid & _T_30; // @[Cache.scala 173:49]
assign _T_32 = _T_31 & io_in_valid; // @[Cache.scala 173:73]
assign _T_33 = metaWay_3_tag == addr_tag; // @[Cache.scala 173:59]
assign _T_34 = metaWay_3_valid & _T_33; // @[Cache.scala 173:49]
assign _T_35 = _T_34 & io_in_valid; // @[Cache.scala 173:73]
assign hitVec = {_T_35,_T_32,_T_29,_T_26}; // @[Cache.scala 173:90]
```

使用Chisel开发的处理器频率符合预期

- 仅针对TLB部分和压缩指令译码部分做少量优化
  - xc7z020-1-clg400 **60MHz**
  - xczu3cg-sfvc784-1-e **200MHz**
  - SMIC 110nm **350MHz**

wer | Methodology | DRC | Timing

Intra-Clock Paths - coreclk_system_top_clk_wiz_0_0 - Setup

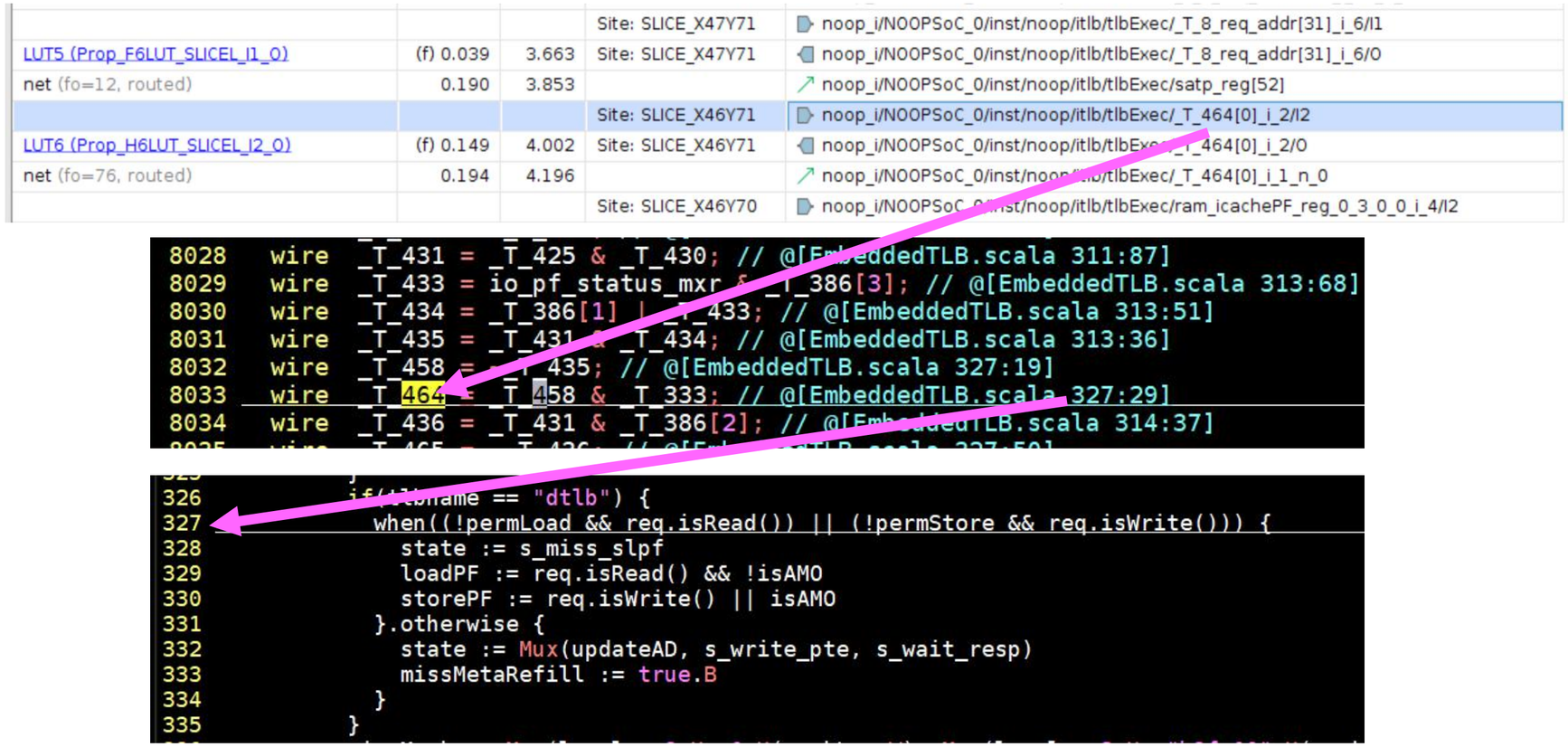
Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	R
Path 1	0.525	14	14	217	system_top_i..._0/CLKARDCLK	system_top_i...RBWRADDR[8]	15.203	4.945	10.258	
Path 10	0.687	14	14	217	system_top_i..._0/CLKARDCLK	system_top_i...RARDADDR[6]	15.165	4.945	10.220	
Path 11	0.687	14	14	217	system_top_i..._0/CLKARDCLK	system_top_i...RARDADDR[6]	15.165	4.945	10.220	
Path 9	0.686	14	14	217	system_top_i..._0/CLKARDCLK	system_top_i...RARDADDR[6]	15.162	4.945	10.217	

Vivado针对Pynq开发板生成的关键路径报告

* 图片来源: http://xilinx.eetrend.com/d6-xilinx/article/2018-11/13898.html

优化关键路径时可追溯到Chisel源码

- 对于名如_T_464的信号, 参照Verilog代码中的注释仍能追溯到其来源
  - 这种信号来源于开发过程中生成的匿名信号(如使用map生成的信号)
  - 已有命名的信号名称会保持原样



处理器在线差分测试仿真验证[1][2]

[1] Yu, EasyDiff: An Effective and Efficient Framework for Processor Verification, CRVF 2019, https://crvf2019.github.io/pdf/14.pdf
[2] 2018年龙芯杯南京大学二队决赛答辩 报告 http://www.nscscc.org/a/wangjie/NSCSCC2018/2018/1010/46.html

处理器在线差分测试仿真验证

- 传统的测试:
  - 生成正确的Trace, 运行处理器并对比Trace, dump波形用于观察

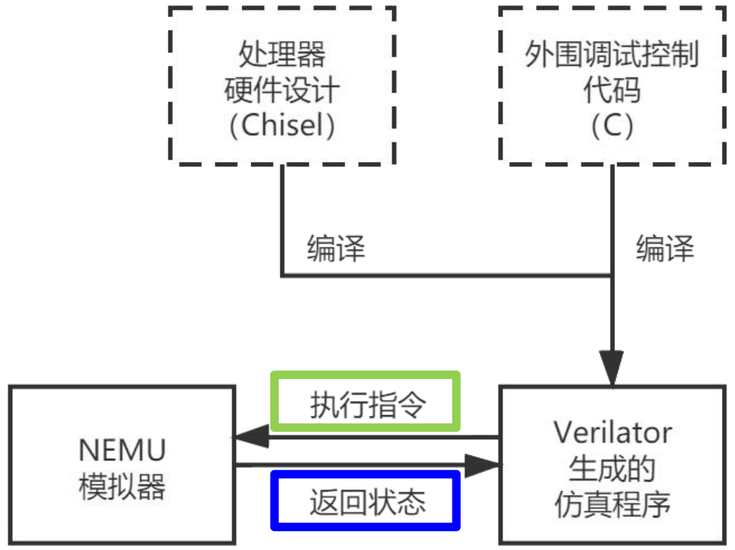
	传统测试方法
存储空间占用	**高**: 波形与Trace需要占用大量空间
验证效率	**较低**: 需要dump波形, 产生大量IO开销, 拖慢验证速度
灵活性	**低**: 无法应对变化的外部输入 (如外部设备状态的变化)
调试难度	**较高**: 波形可读性差

处理器在线差分测试仿真验证

	传统测试方法	我们使用的测试方法
存储空间占用	**高**: 波形与Trace需要占用大量空间	**低**
验证效率	**较低**: 需要dump波形, 产生大量IO开销, 拖慢验证速度	**较高**
灵活性	**低**: 无法应对变化的外部输入 (如外部设备状态的变化)	**高**
调试难度	**较高**: 波形可读性差	**较低**

处理器在线差分测试仿真验证-原理

- 基于指令级在线比对的验证框架
  - 模拟器作为动态链接库连接到Verilator生成的仿真程序中
  - 处理器在指令完成时发出比对请求
  - 在比对结果不一致时报错并输出当前处理器状态



测试框架结构示意

```
while (1) {
    verilator_step();
    nemu_step(1);
    verilator_getregs(&r1);
    nemu_getregs(&r2);
    if (r1 != r2) { abort(); }
}
```

在线比对机制

处理器在线差分测试仿真验证-原理

- 特殊情况的处理
  - 外部中断: 处理器传出中断信息, 模拟器进入相同的处理流程
  - 其他例外: 模拟器支持处理例外, 处理器和模拟器的行为应当一致
  - Memory mapped IO: 处理器标识出这样的访存指令, 在这样的指令提交时从处理器将访存结果复制到模拟器中
    - 例如, 读取(内存地址映射的)时钟计数器
- 这样的处理使得我们的比对机制拥有很高的灵活性

```
if (isMMIO) {
  // MMIO accessing should not be a branch or jump, just +2/+4 to get
  int pc_skip = 0;
  pc_skip += isRVC ? 2 : 4;
  pc_skip += isMultiCommit ? (isRVC2 ? 2 : 4) : 0;
  reg_scala[DIFFTEST_THIS_PC] += pc_skip;
  nemu_this_pc += pc_skip;
  // to skip the checking of an instruction, just copy the reg state t
  ref_difftest_setregs(reg_scala);
  pc_retire_pointer = (pc_retire_pointer+1) % DEBUG_RETIRE_TRACE_SIZE;
  pc_retire_queue[pc_retire_pointer] = this_pc;
  inst_retire_queue[pc_retire_pointer] = this_inst;
  need_copy_pc = 1;
  return 0;
}


if (intrNO) {
  ref_difftest_raise_intr(intrNO);
} else {
  ref_difftest_exec(1);
}
```

测试框架中处理中断/MMIO的部分代码

处理器在线差分测试仿真验证-优势

- 模拟器在仿真过程中同步生成golden trace，提高验证效率
  - **不需要提前生成trace**
  - **可以支持外部中断**
  - 模拟器速度快, 不会对仿真速度造成较大负担
  - 无需dump过多波形, 减少IO开销和磁盘空间占用
- 易于自行修改模拟器实现来辅助验证
  - 例如在访存相关测试时, 针对性地给出特定访存的golden trace
  - 模拟器NEMU是实现了所需功能的最小子集
  - 代码精简, 容易实现
- 直观
  - 直接使用print进行调试, 可读性高
  - 几乎无需观察波形

处理器在线差分测试仿真验证

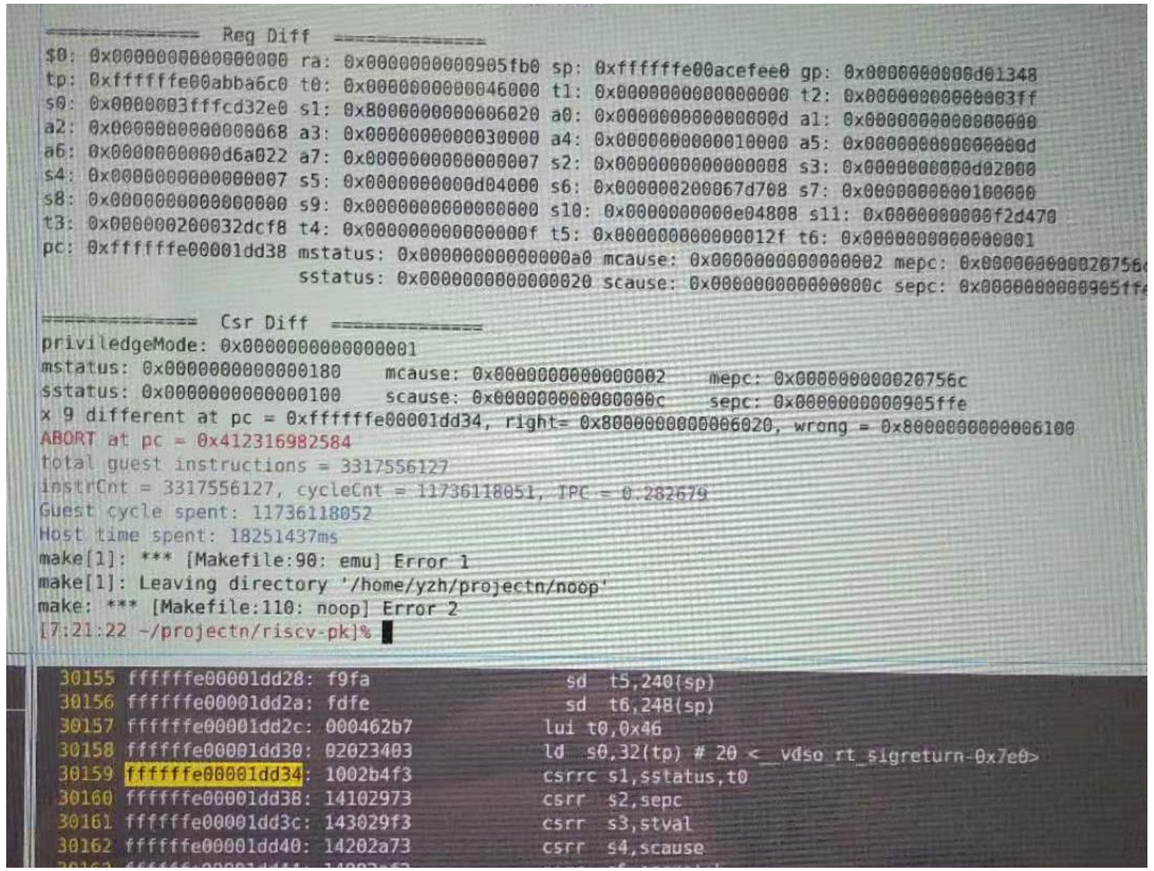
- 测试使用的指令负载包括:

代码片段	性能测试基准程序	操作系统
ProjectN提供的CPU功能测试与性能测试	CoreMark	教学操作系统 XV6
从riscv-test引入的部分测试	Dhrystone	实时操作系统 RT-Thread
riscv-torture 随机指令测试		实时操作系统 FreeRTOS
		Linux Kernel
		Debian

- 依托这一套框架, 在仿真中启动Debian进行验证

处理器在线差分测试仿真验证-示例

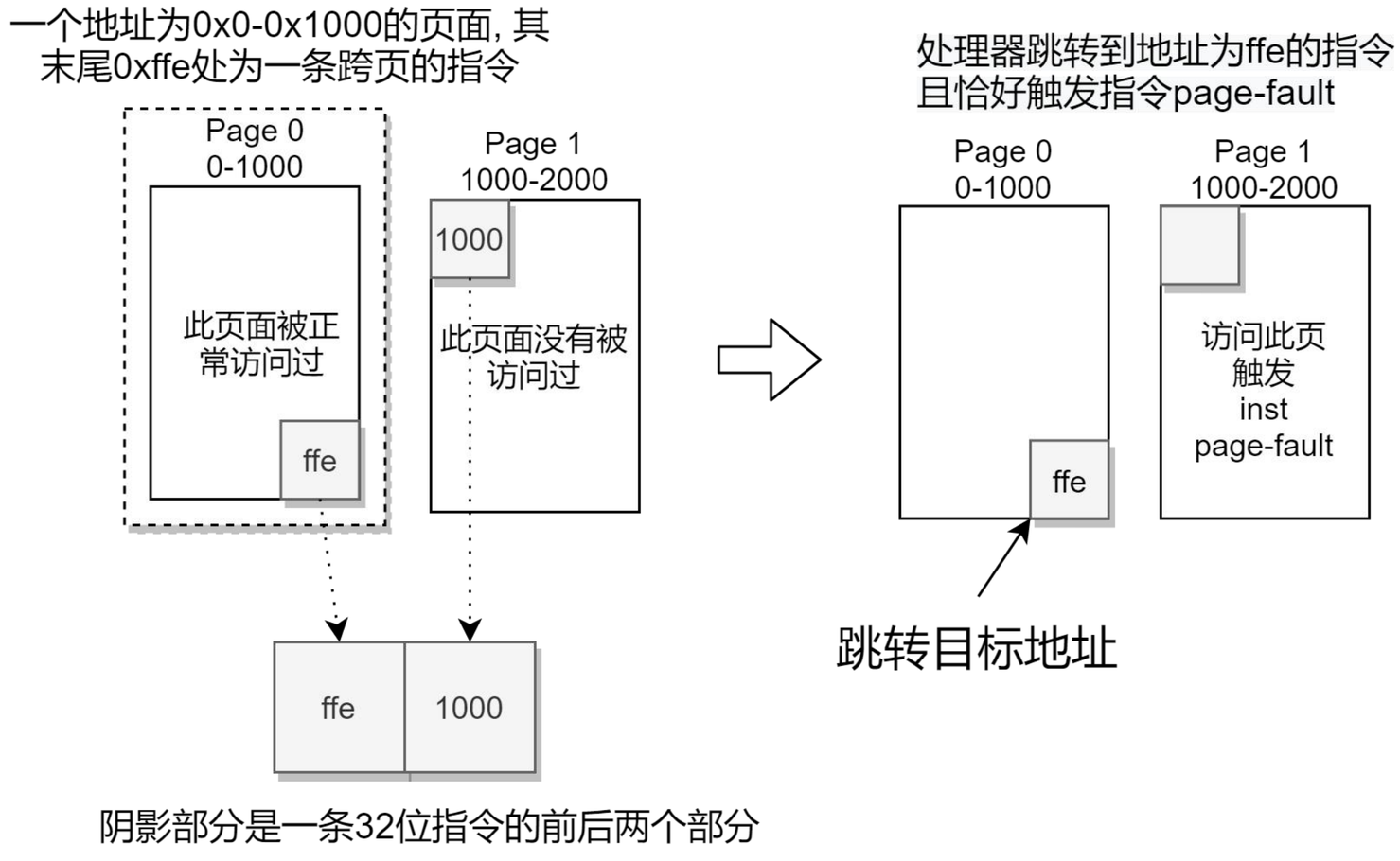
- 使用测试框架定位复杂BUG示例
  - 示例: 跨页指令相关的BUG
  - 在Debian上运行gcc编译复杂程序时发现错误
    - 测试框架在仿真**4小时**, **117亿**个周期, **33亿**条指令后捕捉到了这一错误
    - 暴露出某些很难被检查到的边界问题



测试框架在此处报错

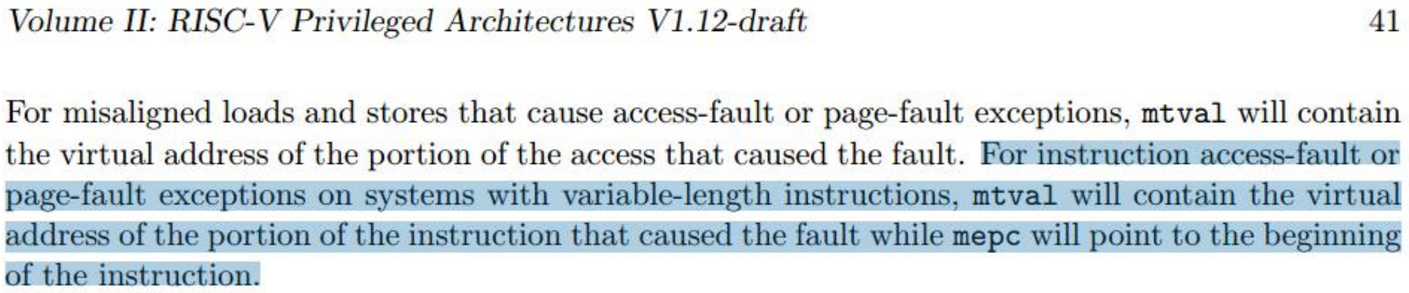
处理器在线差分测试仿真验证-示例

- 示例: 跨页指令相关的BUG



处理器在线差分测试仿真验证-示例

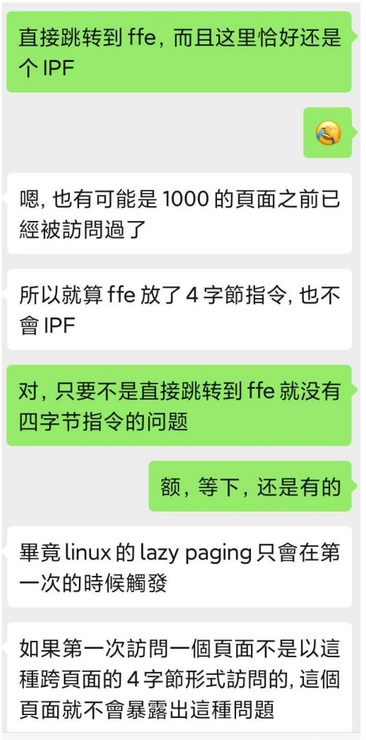
- 使用测试框架定位复杂BUG示例
  - 示例: 跨页指令相关的BUG



Volume II: RISC-V Privileged Architectures V1.12-draft 41

For misaligned loads and stores that cause access-fault or page-fault exceptions, `mtval` will contain the virtual address of the portion of the access that caused the fault. For instruction access-fault or page-fault exceptions on systems with variable-length instructions, `mtval` will contain the virtual address of the portion of the instruction that caused the fault while `mepc` will point to the beginning of the instruction.

指令集手册对这一情况的描述



直接跳转到 ffe，而且这里恰好还是个 IPF

嗯，也有可能是 1000 的頁面之前已經被訪問過了

所以就算 ffe 放了 4 字節指令，也不會 IPF

对，只要不是直接跳转到 ffe 就没有四字节指令的问题

额，等下，还是有的

畢竟 linux 的 lazy paging 只會在第一次的時候觸發

如果第一次訪問一個頁面不是以這種跨頁面的 4 字節形式訪問的，這個頁面就不會暴露出這種問題

开发期间针对这一问题的讨论

- 指令集手册要求在这种指令发生IPF时, 向mepc与mtval寄存器写入不同的值

处理器在线差分测试仿真验证-示例

- 很多bug被光速修复，仅仅留下了git log
  - 基于差分测试框架，在两天内处理了12个启动Linux时的Bug

Commits on Nov 16, 2019

Merge branch 'dev-linux-priv' into 'dev-linux'
AugustusWillisWang committed on 16 Nov 2019

add(riscv64,CSR): set mtval when triggerring page fault in M-mode
AugustusWillisWang committed on 16 Nov 2019

add(riscv64,decode): implement inst wfi as nop
AugustusWillisWang committed on 16 Nov 2019

add(riscv64,exc,CSR): read/write unimplemented csr now triggers an il...
AugustusWillisWang committed on 16 Nov 2019

add(riscv64,intr,CSR): set mtval and stval to 0 except for page fault
AugustusWillisWang committed on 16 Nov 2019

Merge branch 'dev-linux-priv' into 'dev-linux'
AugustusWillisWang committed on 16 Nov 2019

fix(LSU): fix AMO inst decode error
AugustusWillisWang committed on 16 Nov 2019

add(riscv64,CSR): add mstatus.sd bit
AugustusWillisWang committed on 16 Nov 2019

chore(debug): add noop csr display for difftest
AugustusWillisWang committed on 16 Nov 2019

Commits on Nov 17, 2019

Merge branch 'dev-linux' into dev-linux-tlb
Lemover committed on 17 Nov 2019

Merge branch 'dev-linux-fix-tlbpriv' into 'dev-linux'
AugustusWillisWang committed on 17 Nov 2019

fix bug: disable vmEnable at ModeM && add isAMO: loadPF -> storePF wh...
Lemover committed on 17 Nov 2019

noop,fu,CSR: fix wrong mtval/stval for Instruction PF
sashimi-yzh committed on 17 Nov 2019

Merge branch 'dev-linux-priv' into 'dev-linux'
AugustusWillisWang committed on 17 Nov 2019

fix(IDU): ECALL will not influence exceptionVec in IDU
AugustusWillisWang committed on 17 Nov 2019

add(CSR): add exeception priority
AugustusWillisWang committed on 17 Nov 2019

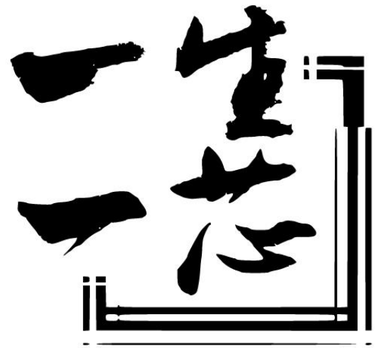
mod(CSR): write satp will now flush pipeline
AugustusWillisWang committed on 17 Nov 2019

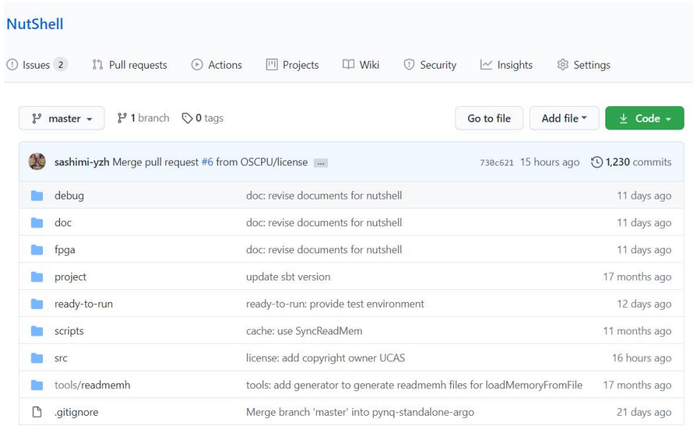
fix(decode): fix decode for invalid inst 0
AugustusWillisWang committed on 17 Nov 2019

chore(debug): add linux to Makefile
AugustusWillisWang committed on 17 Nov 2019

fix(CSR): add inst set 'u' to misa
AugustusWillisWang committed on 17 Nov 2019

开源与致谢





Open Source Chip Project by University (OSCPU)

Let students design their own chips!

https://github.com/OSCPU/NutShell

感谢参与学生指导和为本项目提供帮助的以下人员（姓氏拼音排序）：
蔡晔老师(深圳大学)、李峄工程师(鹏城实验室)、刘彤工程师、唐丹高级工程师、王诲喆博士生、徐易难博士生和余子濠博士生
感谢中国科学院计算技术研究所开源芯片工作组对本项目的指导
感谢南京大学计算机系统教学生态Project-N对本项目的开发和验证提供的帮助

NutShell-本科生设计的可运行Linux的RISC-V芯片

Q & A

中国科学院大学
王华强
2020.7.18