中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES
北京开源芯片研究院
Beijing Institute of Open Source Chip
鹏城实验室
Peng Cheng Laboratory
中国开放指令生态（RISC-V）联盟
China RISC-V Alliance

DiffTest：基于香山处理器的敏捷验证实践

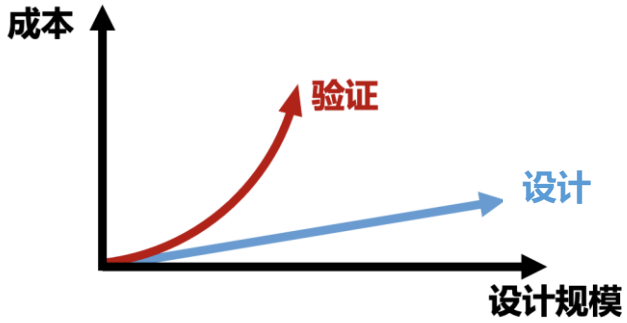
徐易难 王华强 王凯帆 余子濠
中科院计算所

2022年8月26日@第二届RISC-V中国峰会

背景：高性能处理器敏捷开发

- **高性能处理器开发流程长、复杂度高，其中验证是耗时最长的阶段之一**

- **验证墙：处理器设计与验证在敏捷度上持续扩大的差距**
  - 随着设计效率的提高与设计规模扩大，验证将成为处理器开发的瓶颈



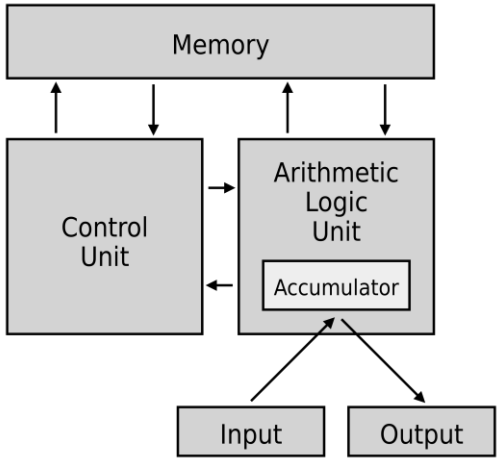
Amdahl's Law：$S_{latency}(s) = \frac{1}{(1-p)+\frac{p}{s}} < \frac{1}{1-p}$

  - **香山：两代架构开发过程中均在验证方面投入了大量的时间与人力**

南湖架构
相比雁栖湖架构
在验证方面
花费了更多时间

背景：处理器功能正确性验证

- **CPU的运行原理：冯诺依曼体系结构**
  - 运算器、控制器、存储器、输入设备、输出设备
  - **存储程序原理**：CPU按约定执行存储器中的指令，直到结束



冯诺依曼结构

- **CPU系统级验证：检查指令集级别行为正确性**

```
while (!end) {
  cpu.exec(1);
}
check_result();
```

(1) End-of-simulation Comparison

```
while (!end) {
  cpu.exec(1);
}
check_trace(N);
```

(2) Trace Comparison

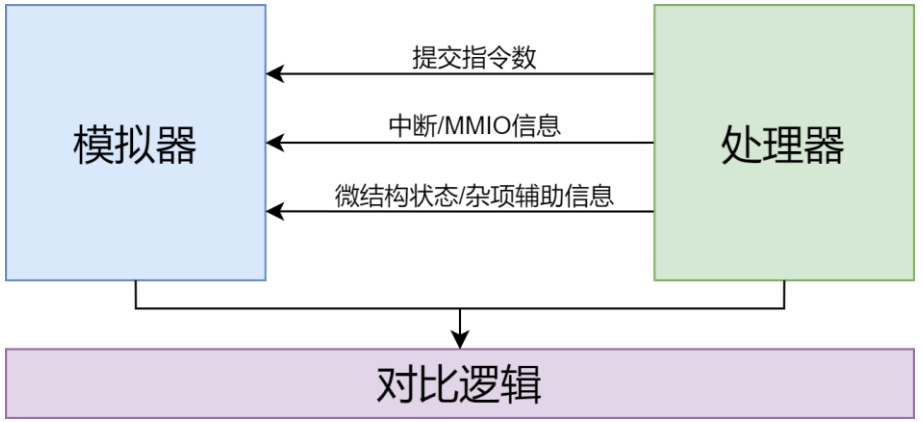
```
while (!end) {
  cpu.exec(1);
  check_trace(1);
}
```

(3) Co-simulation

三种常见的CPU系统级验证方案

DiffTest：指令级在线差分验证框架[1]

- **基本流程**
  - 处理器仿真产生指令提交/其他状态更新
  - 模拟器执行相同的指令
  - 比较两者状态，报错或继续



基本验证框架

- **指令集Golden模型(REF)：NEMU**
  - NEMU: 速度极快的RISC-V解释器[2]
  - 假设NEMU是完全正确的
  - 如何验证NEMU的正确性是另一件事情
  - 也支持Spike作为Golden

```
while (1) {
    icnt = cpu_step();
    nemu_step(icnt);
    r1s = cpu _getregs();
    r2s = nemu_getregs();
    if (r1s != r2s) { abort(); }
}
```

在线对比机制

[1] 王凯帆, 王华强. SMP-MArch-Diff：支持多处理器和RV微结构状态的差分测试方法. https://www.bilibili.com/video/BV1NM4y1T7Hz
[2] 余子濠. NEMU：一个效率接近QEMU的高性能解释器. https://www.bilibili.com/video/BV1Zb4y1k7RJ

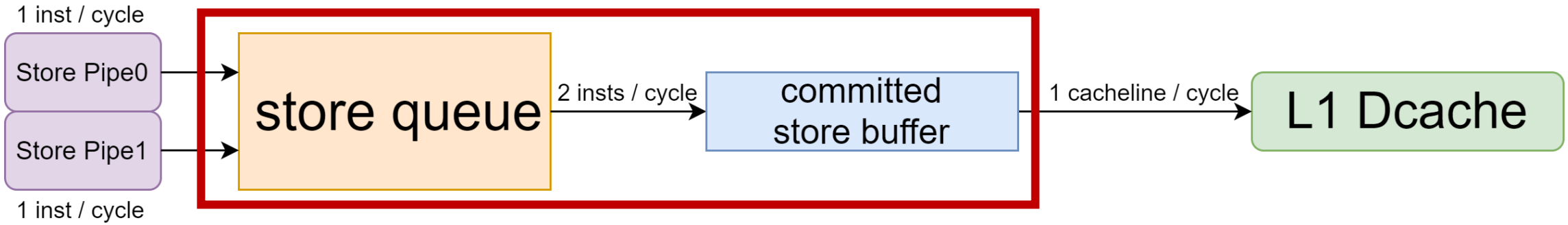
回顾：仅依靠模拟器验证的瓶颈

- 模拟器无法仅靠自己在一些行为上与正确的处理器对齐
- 无法依靠模拟器直接验证处理器的行为

与外部输入相关的行为	与微结构相关的行为	与一致性相关的行为
外部中断	时钟中断	LR/SC
MMIO	Page Fault	多核

回顾：无法对齐的行为分析

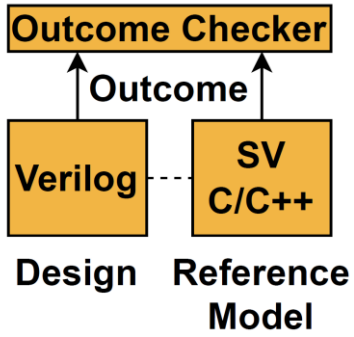
- **案例：缺页异常的产生与否可能与微结构有关**
- 在**没有**使用同步指令的情况下，对**页表**的更改是否对地址映射产生影响？
  - RISC-V 标准没有给出严格约束，是否产生影响**均是可以接受**的行为



会存储一部分已提交 store 的数据
这些数据对页表是**不可见**的

模拟器不具备与设计一致的微结构，如何确定该怎么走？

验证中的非确定性行为



- **验证：用某些方法确认REF与DUT之间的一致性**
  - REF：参考模型 (Reference Model, Golden Model)
  - DUT：待测设计 (Design Under Test)
- **非确定性行为**：REF由于信息缺失而认为DUT有“非确定性“的行为表现
  - 背后的原因：由于REF和DUT在细节上的差异，导致他们的行为存在分歧 (divergence)
  - 导致的结果：REF无法判断什么是“正确“的结果，无法确认DUT的正确性

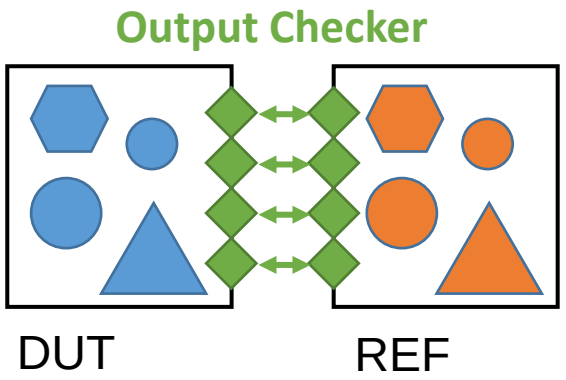
```
class DUT {
  private T num = LFSR64(0xdeadbeaf);

  bool is_good() {
    return (num > 789);
    num.step();
  }
}
```

is_good()

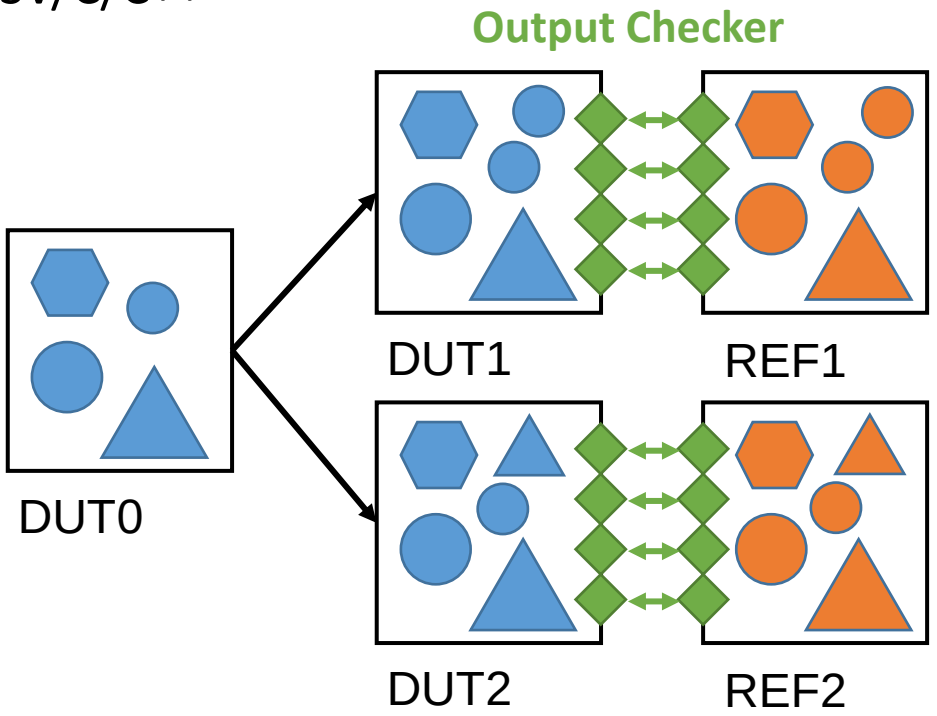
REF

Co-simulation in Conventional Verification

- **Verify the consistency of outputs of DUT and REF**
  - DUT (design-under-test): RTL codes
  - REF (reference model): Golden model, written in SV/C/C++



REF requires impl. details

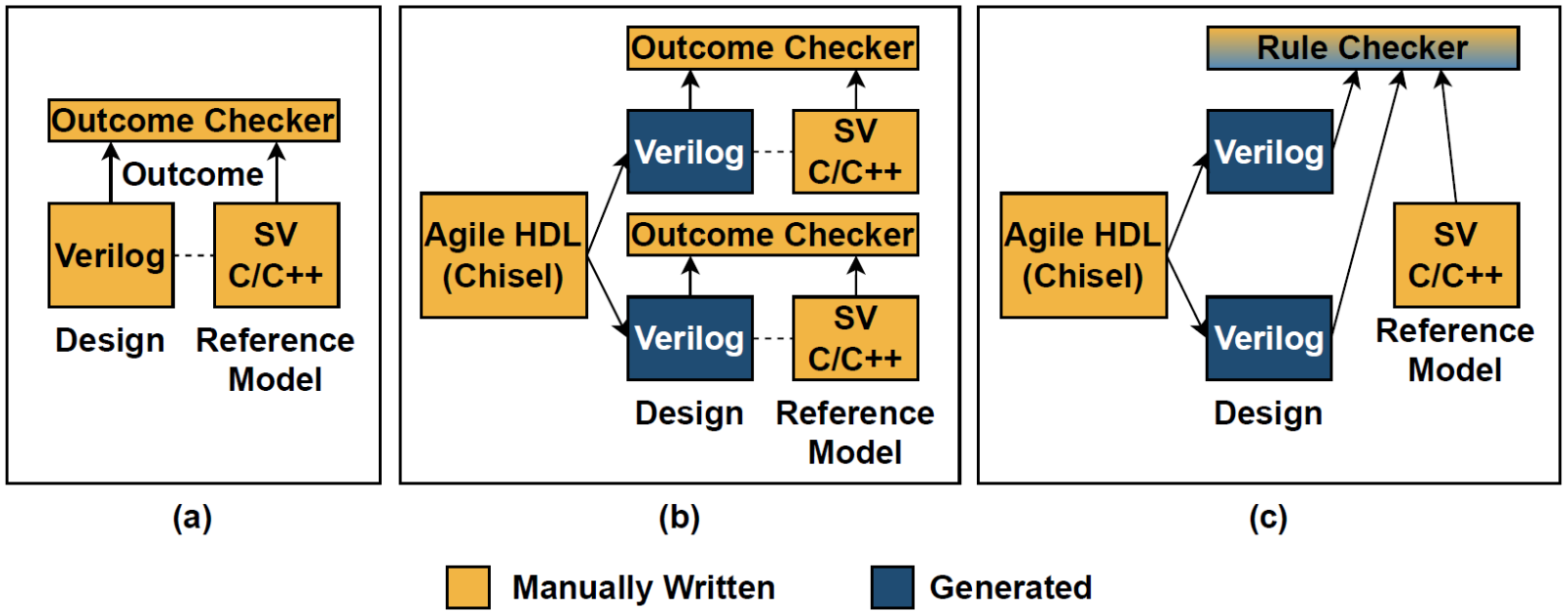


REFs design is slower than DUTs

敏捷设计背景下的验证难题

- **敏捷开发允许更快的设计迭代速度，验证如何跟上？**
  - 如果REF要涉及DUT的设计细节，那么REF和相关验证代码的维护将非常复杂
  - 缺页异常的案例：需要实现微结构对齐的模拟器作为REF，才能确保获得正确的结果

- **理想：针对不同的设计，复用验证逻辑和代码**

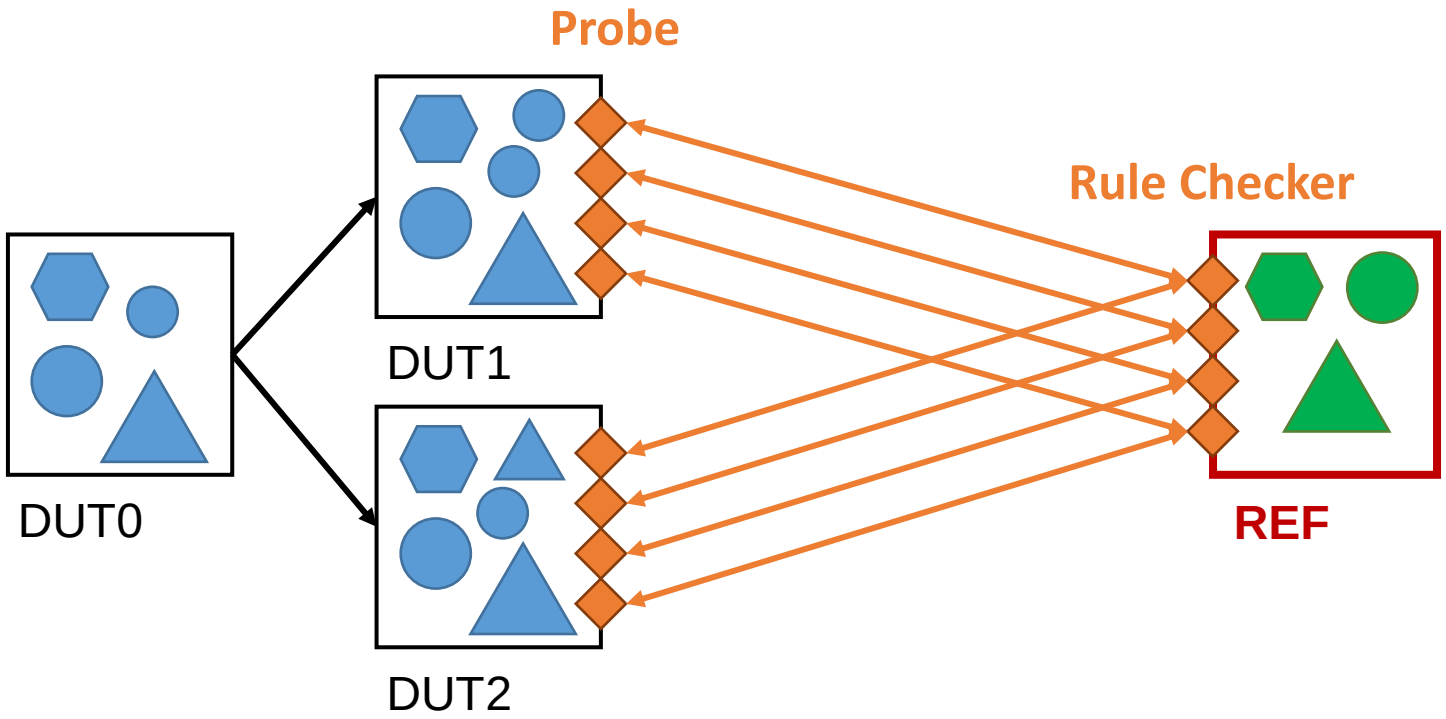


Diff-Rule based Agile Verification (DRAV)

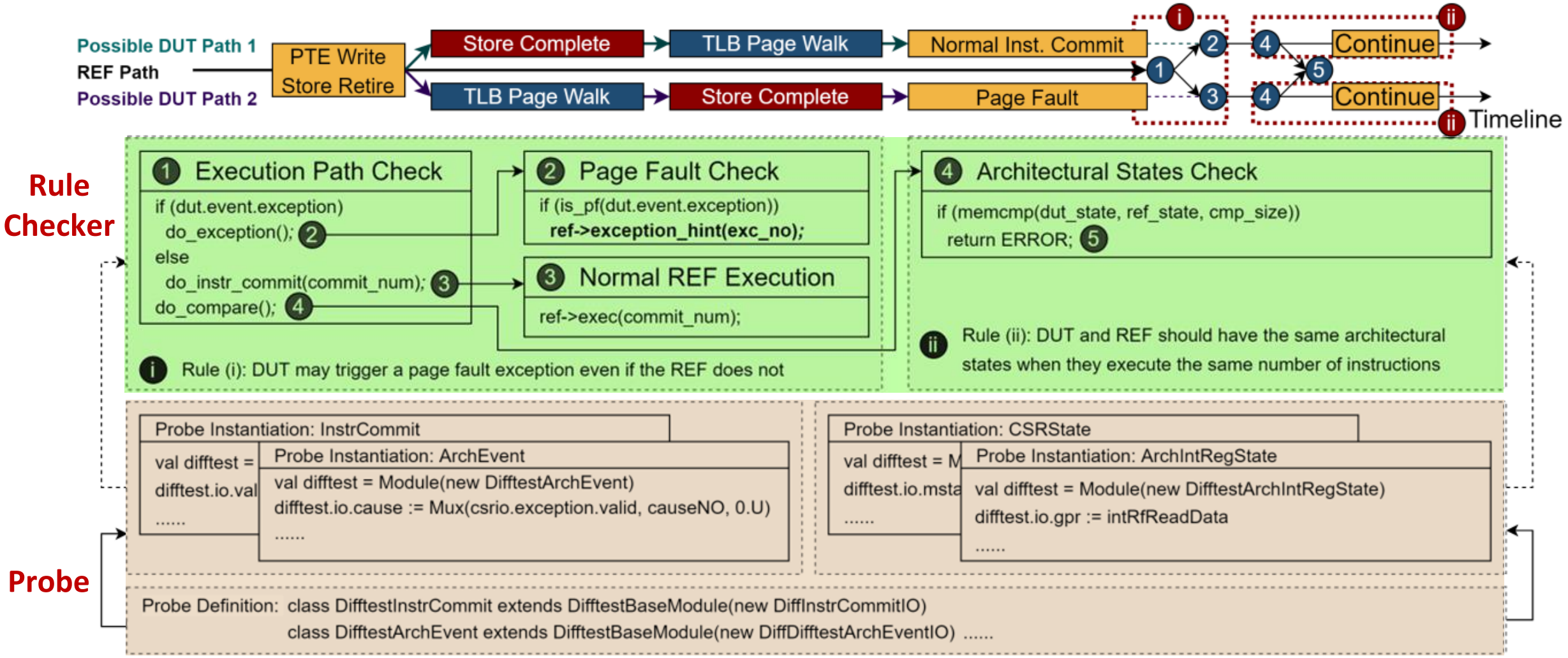
- What is correctness?
- What is a Golden REF?
- Insights
  - Specifications are eventual golden REFs.
  - A given a design specification can lead to diverse implementations.
- How to describe specifications?
  - Natural Languages (NLs), SAIL model, SystemVerilog (SV), ……

Diff-Rule based Agile Verification (DRAV)

- **Use rules to describe behaviors defined by spec**
  - **Example #1**: DUT and REF should have the same architectural states then they execute the same number of instructions.
  - **Example #2:** A page fault exception can be triggered in DUT even if it does not occur in REF.

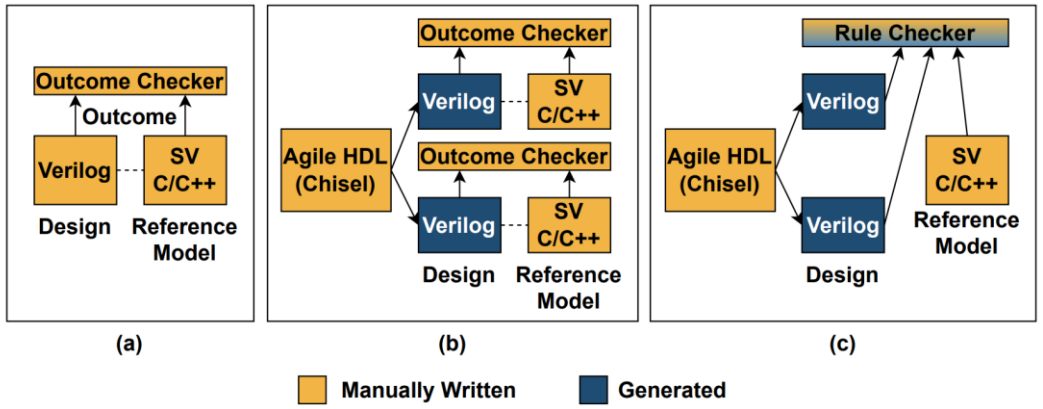


Diff-Rule based Agile Verification (DRAV)



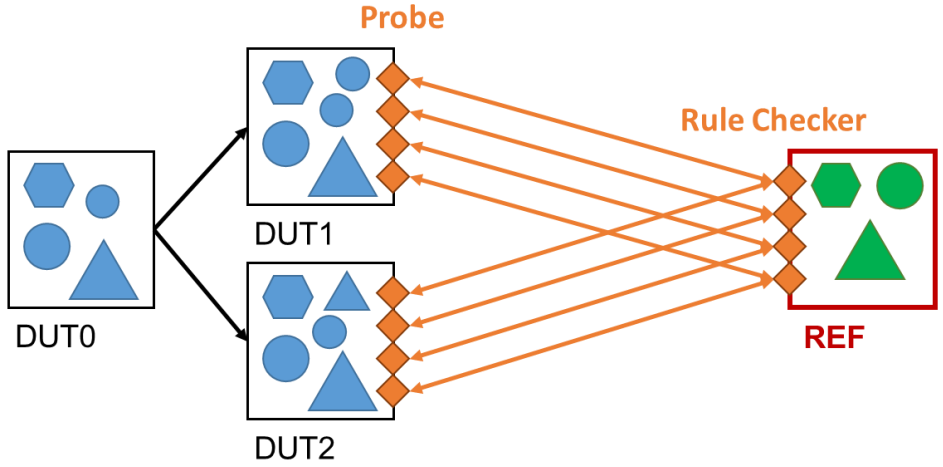
DiffTest：通用的在线差分验证框架

- **基本流程**
  - 处理器仿真产生指令提交/其他状态更新
  - 模拟器执行相同的指令
  - 比较两者状态，报错或继续



不同的验证框架搭建方式

- **DRAV：如何更好地支持 RISC-V 处理器敏捷验证**
  - **更多的处理器、更复杂的场景、更快速的设计迭代**
  - 使用 Diff-rule 描述设计规范所允许的行为
  - 使用 Probe 完成微结构信息的传递



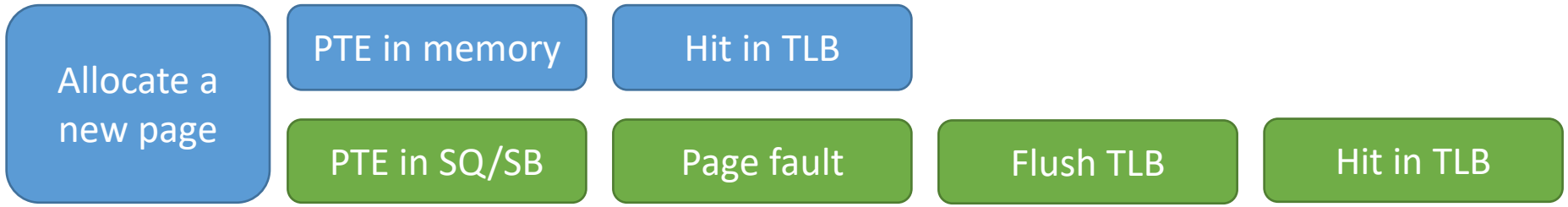
降低验证框架与 REF 的复用成本

Diff-rule：定位设计规范中的非确定性

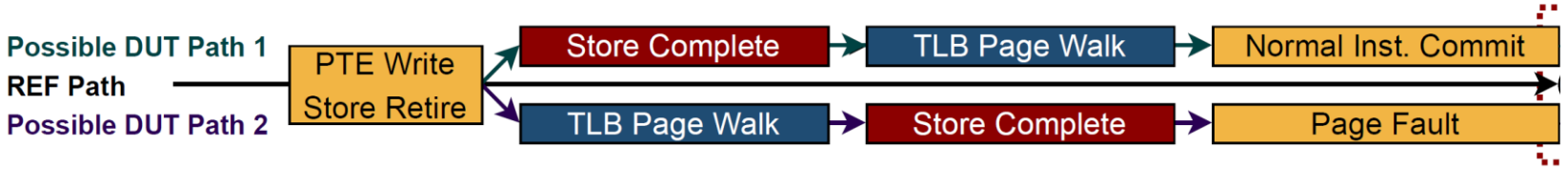
- **Diff-rule：在特定的场景下，允许设计具有多种合法行为**
- **关键：定位设计规范中的非确定性，并在验证框架中以一定规则要求放松检查**
- **非确定性：以 RISC-V 处理器为例**
  - 推测的地址翻译 Speculative address translation
  - 缓存层次与多核 cache hierarchy and multi-core scenarios
  - 其他更多的内容：中断、LR/SC指令、硬件性能计数器等

来源①：推测的地址翻译

Linux: performance improvement by lazily executing sfence.vma



Hardware: performance improvement by speculative address translation



- **Linux 在分配新页表后，会选择不冲刷 TLB 缓存，造成不确定的缺页事件**
  - RISC-V 允许 TLB 的推测访问，且允许缓存无效项，实现性能优化
  - Linux 通过减少 TLB 冲刷，推测认为 TLB 能够拿到有效 PTE，实现性能优化

- **从验证角度看，REF如何确定此时的CPU是否应当触发缺页异常?**

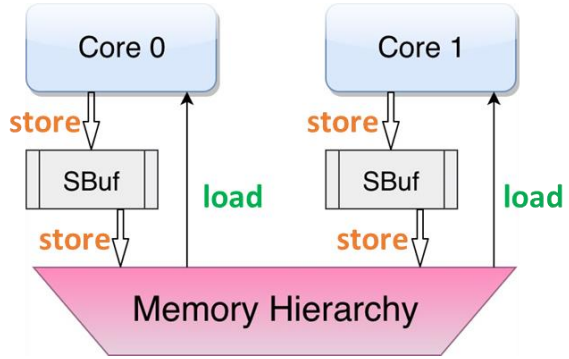
来源①：推测的地址翻译

- **解决方案：维护微结构体系结构状态**
  - 需要至少维护 Store Queue, Store Buffer, TLB, PTW, Cache等大量微体系结构状态
  - 维护成本过高，无法满足敏捷开发快速迭代的要求

- **DiffTest：仅维护 RISC-V 体系结构状态，允许 DUT 发生缺页异常**
  - 同时提示Warning，并在发生异常后，检查两者状态是否一致、
  - 可实现更多的检查，以确保行为符合RISC-V规范，如维护sfence.vma后的store请求历史
  - 维护成本低，可实现；设计行为符合规范，但无法确认设计行为与设计意图一致

来源②：缓存层次与多核

- Load 指令允许本地 store buffer + 全局 global memory 的存在



- **REF 如何确定一个核的写入何时被另一个核看见**
  - 多核存在exponential interleaving space of concurrent memory accesses
  - 完整的多核REF搭建复杂度极高，涉及到大量的微结构行为细节

- **DiffTest：使用单核 REF + Global Memory 实现多核验证**
  - Global Memory: 维护全局内存状态，基于 store buffer 请求握手时刻进行数据更新
  - Diff-rule：当Load数据与单核REF不一致时，允许从Global Memory中获取新数据

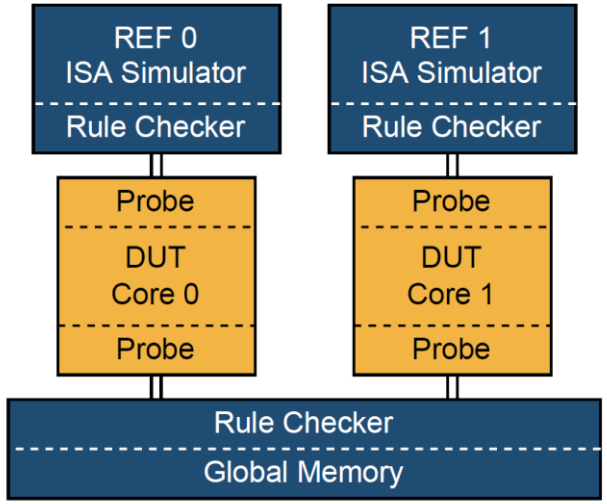
更多的非确定性行为来源

- **中断：可标记在任意一条指令上，REF无法确定准确位置**
  - Diff-rule：允许DUT发生中断，检查中断后的状态是否正确
- **LR/SC指令：SC可能因为特定的微结构状态导致失败**
  - Diff-rule：允许SC指令失败，检查失败后的状态是否正确
- **指令融合：CPU可能将多条指令融合成一个，并只提交一次**
  - Diff-rule：允许DUT合并提交多条指令，检查执行完成后的结果是否正确
- **硬件性能计数器：大量微结构相关的性能计数器**
  - Diff-rule：允许在比较时跳过一些指令，将这些指令的寄存器写回结果更新至REF

DiffTest：对验证目标的清晰定义

- **最佳的 RISC-V 处理器系统级 co-simulation 框架**
  - 在这一场景下，验证关注的是指令集层面的兼容性
  - **核心问题：对指令集层面非确定性行为的刻画**

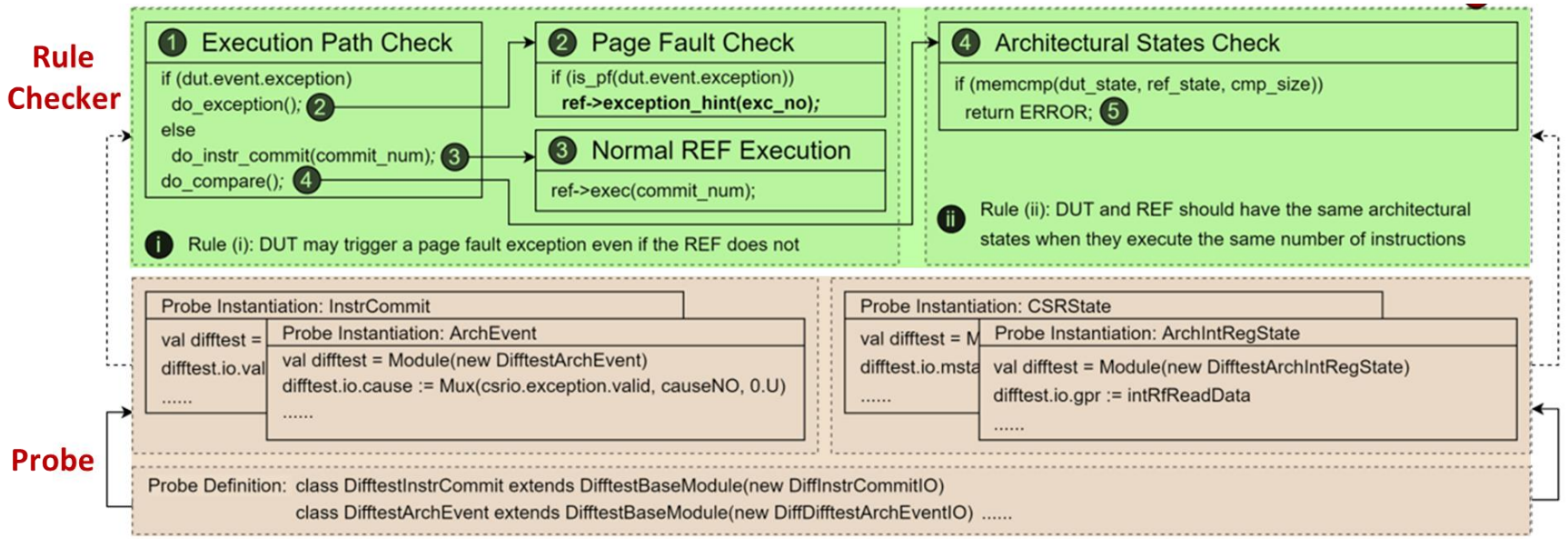
- **配合更丰富的验证工作，完成对处理器的多层次验证**
  - BPUTester：关注分支预测部件的性能情况
  - 浮点单元测试：关注浮点运算部件的行为正确性
  - TL-Test：关注Cache的行为正确性
  - ……



DiffTest架构

Probe：沟通设计与验证之间的桥梁

- **设计变动频繁，如何实现验证框架的可复用性**
  - 验证接口变动、设计参数变动
- **关键：Chisel等高层次HDL支持code-generation，可以自由嵌入验证代码**
  - 结构化的bundle定义，自动化的函数生成，参数化的设计与验证代码 → 通用的验证框架
- **DiffTest = diff-rules + probes**



DiffTest：极低的性能开销与良好的可扩展性

- C/C++代码实现的REF和diff-rules
- 基于DPI-C完成信息传递

- 基于Diff-rule扩展验证能力
  - 多层次的diff-rule设计
  - 可复用的验证框架

- 基于Probe支持更丰富的调试能力
  - ArchDB：自动生成的SQL

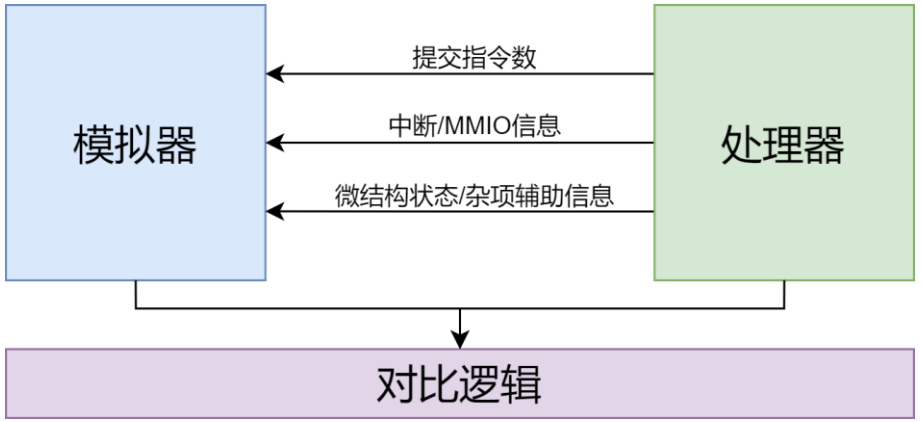
```
[    0.000000] OF: fdt: Ignoring memory range 0x80000000 - 0x80200000
[    0.000000] Linux version 4.18.0-00048-g9be229d2ec2c-dirty (wkf@xiangshan-06) (gcc version 9.2.0 (GCC)) #159 SMP Sun
[    0.000000] bootconsole [early0] enabled
[    0.000000] Initial ramdisk at: 0x(____ptrval____) (23552 bytes)
[    0.000000] Zone ranges:
[    0.000000]   DMA32    empty
[    0.000000]   Normal   [mem 0x0000000080200000-0x0000000081ffffff]
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x0000000080200000-0x0000000081ffffff]
[    0.000000] Initmem setup node 0 [mem 0x0000000080200000-0x0000000081ffffff]
[    0.000000] Cannot allocate SWIOTLB buffer
[    0.000000] elf_hwcap is 0x112d
[    0.000000] percpu: Embedded 11 pages/cpu @(____ptrval____) s15072 r0 d29984 u45056
[    0.000000] Built 1 zonelists, mobility grouping on.  Total pages: 7575
[    0.000000] Kernel command line: root=/dev/mmcblk0 rootfstype=ext4 ro rootwait earlycon
[    0.000000] Dentry cache hash table entries: 4096 (order: 3, 32768 bytes)
[    0.000000] Inode-cache hash table entries: 2048 (order: 2, 16384 bytes)
[    0.000000] Sorting __ex_table...
[    0.000000] Memory: 28936K/30720K available (780K kernel code, 78K rwdata, 109K rodata, 110K init, 100K bss, 1784K r
[    0.000000] SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=2, Nodes=1
[    0.000000] Hierarchical RCU implementation.
[    0.000000] NR_IRQS: 0, nr_irqs: 0, preallocated irqs: 0
[    0.000000] clocksource: riscv_clocksource: mask: 0xffffffffffffffff max_cycles: 0x1d854df40, max_idle_ns: 352636161
[    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: 4096 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] Hierarchical SRCU implementation.
[    0.000000] smp: Bringing up secondary CPUs ...
[    0.000000] smp: Brought up 1 node, 2 CPUs
[    0.000000] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 19112604462750000 ns
[    0.000000] clocksource: Switched to clocksource riscv_clocksource
[    0.000000] Unpacking initramfs...
[    0.000000] workingset: timestamp_bits=62 max_order=13 bucket_order=0
[    0.000000] random: get_random_bytes called from 0xffffffff8001f912 with crng_init=0
[    0.000000] Freeing unused kernel memory: 108K
[    0.000000] This architecture does not have kernel memory protection.
Hello, RISC-V World!
HIT GOOD TRAP at pc = 0x7f80034ee6
total guest instructions = 5,626,704
```

双核 Linux Kernel 启动

Overhead: <1%

DiffTest的使用方法

- **工作流程拆解**
  - 处理器仿真产生指令提交/其他状态更新
  - 模拟器执行相同的指令
  - 比较两者状态，决定报错或继续



基本验证框架

仿真框架

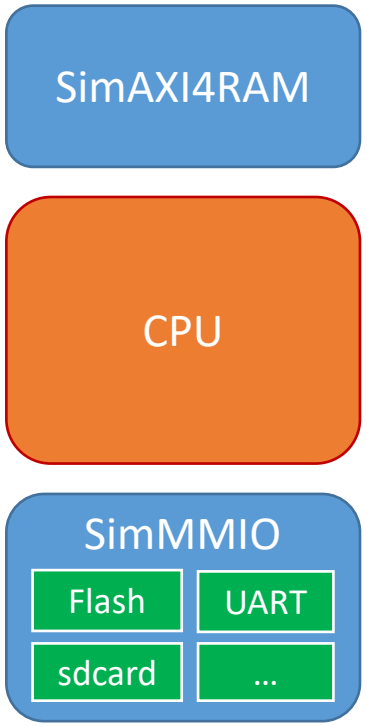
抓取CPU信息

REF(NEMU, Spike)运行

结果对比

DiffTest流程①：仿真框架

- 支持CPU全系统仿真，包含RAM、UART、SD卡等外设
  - SOC中包含一个SimMMIO，并通过它来完成外设的交互，实现上是用了DPIC




```
import "DPI-C" function void ram_write_helper
(
  input  longint    wIdx,
  input  longint    wdata,
  input  longint    wmask,
  input  bit        wen
);

import "DPI-C" function longint ram_read_helper
(
  input  bit        en,
  input  longint    rIdx
);

module RAMHelper(
  input         clk,
  input         en,
  input  [63:0] rIdx,
  output [63:0] rdata,
  input  [63:0] wIdx,
  input  [63:0] wdata,
  input  [63:0] wmask,
  input         wen
);

  assign rdata = ram_read_helper(en, rIdx);

  always @(posedge clk) begin
    ram_write_helper(wIdx, wdata, wmask, wen && en);
  end

endmodule
```

硬件侧 (ram.v)

```
extern "C" uint64_t ram_read_helper(uint8_t en, uint64_t rIdx) {
  if (!ram)
    return 0;
  if (en && rIdx >= EMU_RAM_SIZE / sizeof(uint64_t)) {
    rIdx %= EMU_RAM_SIZE / sizeof(uint64_t);
  }
  pthread_mutex_lock(&ram_mutex);
  uint64_t rdata = (en) ? ram[rIdx] : 0;
  pthread_mutex_unlock(&ram_mutex);
  return rdata;
}

extern "C" void ram_write_helper(uint64_t wIdx, uint64_t wdata, uint64_t wmask, uint8_t wen) {
  if (wen && ram) {
    if (wIdx >= EMU_RAM_SIZE / sizeof(uint64_t)) {
      printf("ERROR: ram wIdx = 0x%lx out of bound!\n", wIdx);
      assert(wIdx < EMU_RAM_SIZE / sizeof(uint64_t));
    }
    pthread_mutex_lock(&ram_mutex);
    ram[wIdx] = (ram[wIdx] & ~wmask) | (wdata & wmask);
    pthread_mutex_unlock(&ram_mutex);
  }
}
```

仿真框架 (ram.cpp)

DiffTest流程②：抓取CPU信息

- Chisel中定义并拉取原始信息
- 将原始信息存进C++的数据结构
- 使用C++中的数据完成后续验证流程

```scala
class DifftestBaseModule[T <: DifftestBundle](gen: T) extends DifftestModule[T] {
  val io = IO(gen)
  instantiate()
}

class DifftestArchEvent extends DifftestBaseModule(new DiffArchEventIO)
class DifftestBasicInstrCommit extends DifftestBaseModule(new DiffBasicInstrCommitIO)
class DifftestInstrCommit extends DifftestBaseModule(new DiffInstrCommitIO)
class DifftestBasicTrapEvent extends DifftestBaseModule(new DiffBasicTrapEventIO)
class DifftestTrapEvent extends DifftestBaseModule(new DiffTrapEventIO)
class DifftestCSRState extends DifftestBaseModule(new DiffCSRStateIO)
class DifftestDebugMode extends DifftestBaseModule(new DiffDebugModeIO)
class DifftestIntWriteback extends DifftestBaseModule(new DiffIntWritebackIO)
class DifftestFpWriteback extends DifftestBaseModule(new DiffFpWritebackIO)
class DifftestArchIntRegState extends DifftestBaseModule(new DiffArchIntRegStateIO)
class DifftestArchFpRegState extends DifftestBaseModule(new DiffArchFpRegStateIO)
class DifftestSbufferEvent extends DifftestBaseModule(new DiffSbufferEventIO)
class DifftestStoreEvent extends DifftestBaseModule(new DiffStoreEventIO)
class DifftestLoadEvent extends DifftestBaseModule(new DiffLoadEventIO)
class DifftestAtomicEvent extends DifftestBaseModule(new DiffAtomicEventIO)
class DifftestPtwEvent extends DifftestBaseModule(new DiffPtwEventIO)
class DifftestRefillEvent extends DifftestBaseModule(new DiffRefillEventIO)
class DifftestLrScEvent extends DifftestBaseModule(new DiffLrScEventIO)
class DifftestRunaheadEvent extends DifftestBaseModule(new DiffRunaheadEventIO)
```

传入一些bundle信息
自动生成DPIC调用

硬件侧 (Difftest.scala)

```scala
INTERFACE_BASIC_TRAP_EVENT {
  RETURN_NO_NULL
  auto packet = difftest[coreid]->get_trap_event();
  packet->valid    = valid;
  packet->cycleCnt = cycleCnt;
  packet->instrCnt = instrCnt;
  packet->hasWFI   = hasWFI;
}

INTERFACE_ARCH_EVENT {
  RETURN_NO_NULL
  auto packet = difftest[coreid]->get_arch_event();
  packet->interrupt = intrNo;
  packet->exception = cause;
  packet->exceptionPC = exceptionPC;
  packet->exceptionInst = exceptionInst;
}

INTERFACE_BASIC_INSTR_COMMIT {
  RETURN_NO_NULL
  auto packet = difftest[coreid]->get_instr_commit(index);
  packet->valid    = valid;
  if (packet->valid) {
    packet->skip     = skip;
    packet->rfwen    = rfwen;
  }
}
```

DPIC函数体将信号
存到C的struct中
（暂时是人工写的）

中间层 (interface.cpp)

```scala
typedef struct {
  uint8_t  valid = 0;
  uint8_t  is_load;
  uint8_t  need_wait;
  uint64_t pc;
  uint64_t oracle_vaddr;
} run_ahead_memdep_pred_t;

typedef struct {
  uint64_t gpr[DIFFTEST_MAX_PRF_SIZE];
  uint64_t fpr[DIFFTEST_MAX_PRF_SIZE];
} physical_reg_state_t;

typedef struct {
  trap_event_t      trap;
  arch_event_t      event;
  instr_commit_t    commit[DIFFTEST_COMMIT_WIDTH];
  arch_reg_state_t  regs;
  arch_csr_state_t  csr;
  debug_mode_t      dmregs;
  sbuffer_state_t   sbuffer[DIFFTEST_SBUFFER_RESP_WIDTH];
  store_event_t     store[DIFFTEST_STORE_WIDTH];
  load_event_t      load[DIFFTEST_COMMIT_WIDTH];
  atomic_event_t    atomic;
  ptw_event_t       ptw;
  refill_event_t    refill;
  lr_sc_evevnt_t    lrsc;
  run_ahead_event_t runahead[DIFFTEST_RUNAHEAD_WIDTH];
  run_ahead_commit_event_t runahead_commit[DIFFTEST_RUNAHEAD_WIDTH];
  run_ahead_redirect_event_t runahead_redirect;
  run_ahead_memdep_pred_t runahead_memdep_pred[DIFFTEST_RUNAHEAD_WIDTH];
  physical_reg_state_t pregs;
} difftest_core_state_t;
```

使用struct信息
完成验证

验证侧 (difftest.cpp)

DiffTest流程③：REF运行

- 基本原理：CPU执行一条指令，NEMU也执行一条指令

```
134   num_commit = 0; // reset num_commit this cycle to 0
135   // interrupt has the highest priority
136   if (dut.event.interrupt) {
137     dut.csr.this_pc = dut.event.exceptionPC;
138     do_interrupt();
139   } else if (dut.event.exception) {
140     // We ignored instrAddrMisaligned exception (0) for better debug interfac
141     // XiangShan should always support RVC, so instrAddrMisaligned will never
142     // TODO: update NEMU, for now, NEMU will update pc when exception happen
143     dut.csr.this_pc = dut.event.exceptionPC;
144     do_exception();
145   } else {
146     // TODO: is this else necessary?
147     for (int i = 0; i < DIFFTEST_COMMIT_WIDTH && dut.commit[i].valid; i++) {
148       do_instr_commit(i);
149       dut.commit[i].valid = 0;
150       num_commit++;
151       // TODO: let do_instr_commit return number of instructions in this uop
152       if (dut.commit[i].fused) {
153         num_commit++;
154       }
155     }
156   }
```

对每一条提交的指令，

调用do_instr_commit

让REF同样执行一条指令

```
274   // single step exec
275   proxy->exec(1);
276   // when there's a fused instruction
277   if (dut.commit[i].fused) {
278     proxy->exec(1);
279   }
```

```
258   // MMIO accessing should not be a branch or jump, just +2/+4 to get the next pc
259   // to skip the checking of an instruction, just copy the reg state to reference design
260   if (dut.commit[i].skip || (DEBUG_MODE_SKIP(dut.commit[i].valid, dut.commit[i].pc, dut.commit[i].inst))) {
261     proxy->regcpy(ref_regs_ptr, REF_TO_DIFFTEST);
262     ref.csr.this_pc += dut.commit[i].isRVC ? 2 : 4;
263     if (realWen) {
264       // We use the physical register file to get wdata
265       // TODO: what if skip with fpwen?
266       ref_regs_ptr[dut.commit[i].wdest] = get_commit_data(i);
267       // printf("Debug Mode? %x is ls? %x\n", DEBUG_MEM_REGION(dut.commit[i].valid, dut.commit[i].pc), IS_LOAD_S
268       // printf("skip %x %x %x %x %x\n", dut.commit[i].pc, dut.commit[i].inst, get_commit_data(i), dut.commit[i]
269     }
270     proxy->regcpy(ref_regs_ptr, DIFFTEST_TO_REF);
271     return;
272   }
```

还有一些例化情况，比如NEMU无法模拟的外设访问等

CPU的状态会被拷贝给NEMU

DiffTest流程④：结果对比

- 对比通用寄存器和CSR的值

```c
typedef struct {
  uint64_t gpr[32];
  uint64_t fpr[32];
} arch_reg_state_t;

typedef struct __attribute__((packed)) {
  uint64_t this_pc;
  uint64_t mstatus;
  uint64_t mcause;
  uint64_t mepc;
  uint64_t sstatus;
  uint64_t scause;
  uint64_t sepc;
  uint64_t satp;
  uint64_t mip;
  uint64_t mie;
  uint64_t mscratch;
  uint64_t sscratch;
  uint64_t mideleg;
  uint64_t medeleg;
  uint64_t mtval;
  uint64_t stval;
  uint64_t mtvec;
  uint64_t stvec;
  uint64_t priviledgeMode;
} arch_csr_state_t;
```

对比两侧的寄存器值

```c
if (memcmp(dut_regs_ptr, ref_regs_ptr, DIFFTEST_NR_REG * sizeof(uint64_t))) {
  display();
  for (int i = 0; i < DIFFTEST_NR_REG; i ++) {
    if (dut_regs_ptr[i] != ref_regs_ptr[i]) {
      printf("%7s different at pc = 0x%010lx, right= 0x%016lx, wrong = 0x%016lx\n",
          reg_name[i], ref.csr.this_pc, ref_regs_ptr[i], dut_regs_ptr[i]);
    }
  }
  return 1;
}

return 0;
```

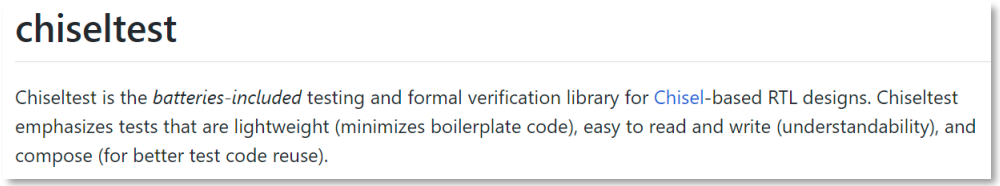
```c
int Difftest::do_store_check() {
  for (int i = 0; i < DIFFTEST_STORE_WIDTH; i++) {
    if (!dut.store[i].valid) {
      return 0;
    }
    auto addr = dut.store[i].addr;
    auto data = dut.store[i].data;
    auto mask = dut.store[i].mask;
    if (proxy->store_commit(&addr, &data, &mask)) {
      display();
      printf("Mismatch for store commits %d: \n", i);
      printf("  REF commits addr 0x%lx, data 0x%lx, mask 0x%x\n", addr, data, mask);
      printf("  DUT commits addr 0x%lx, data 0x%lx, mask 0x%x\n",
        dut.store[i].addr, dut.store[i].data, dut.store[i].mask);
      return 1;
    }
    dut.store[i].valid = 0;
  }
  return 0;
}
```

还有更多对比，

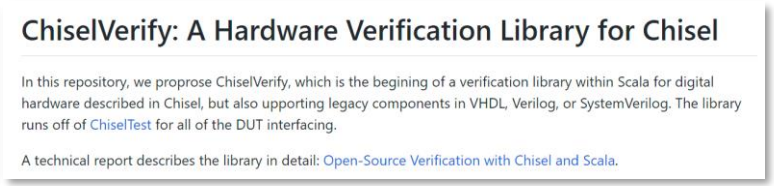
比如STORE的结果

DiffTest-NG: Next-Generation DiffTest

- **面向高层次设计的通用验证流程与编程框架：Verification with High-level HDLs**



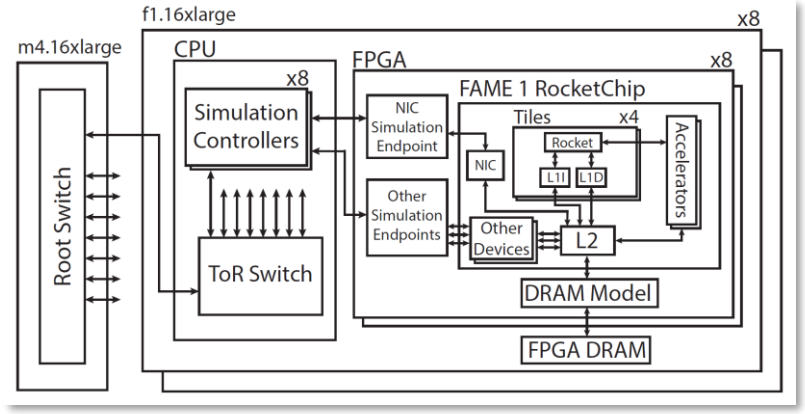
ChiselTest (UCB)



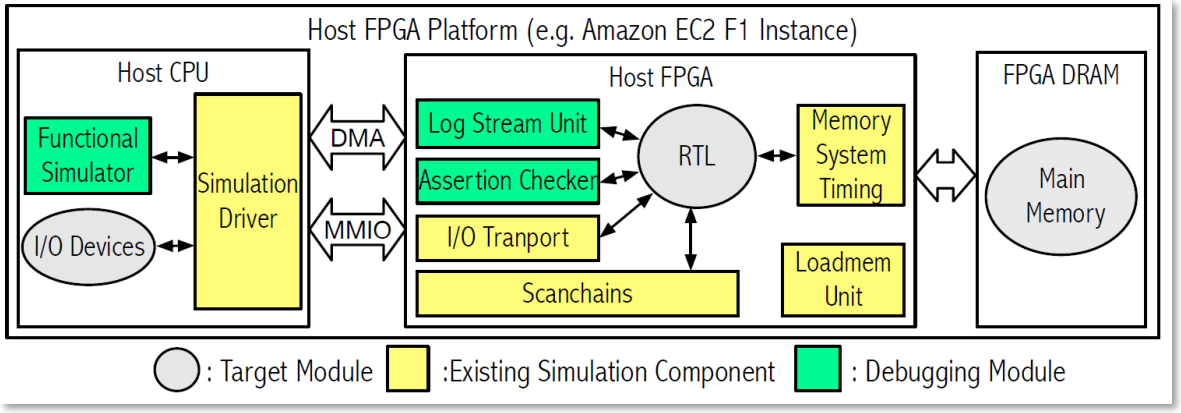
ChiselVerify (DTU)

Verif (UCB)

- **FPGA加速的验证流程： DiffTest (and Debugging Plugins) on FPGA**



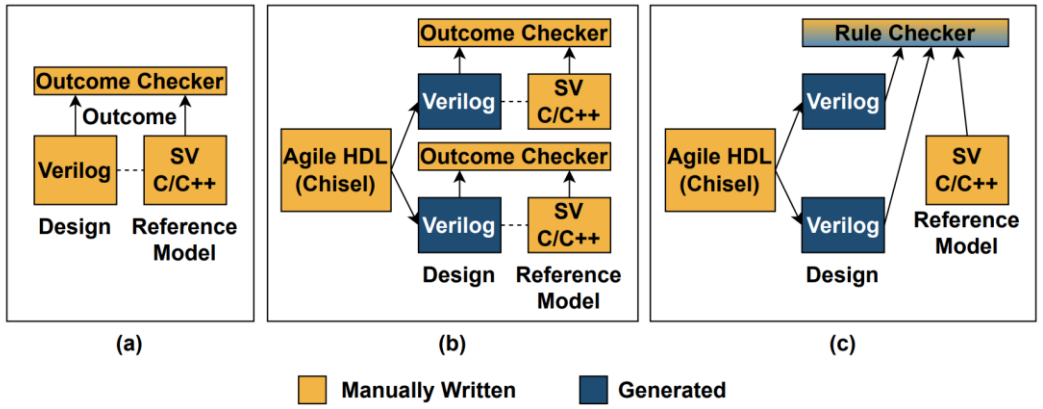
FireSim[ISCA'18]



DESSERT[FPL'18]

DiffTest：通用的在线差分验证框架

- **基本流程**
  - 处理器仿真产生指令提交/其他状态更新
  - 模拟器执行相同的指令
  - 比较两者状态，报错或继续



不同的验证框架搭建方式

- **DRAV：如何更好地支持 RISC-V 处理器敏捷验证**
  - **更多的处理器、更复杂的场景、更快速的设计迭代**
  - 使用 Diff-rule 描述设计规范所允许的行为
  - 使用 Probe 完成微结构信息的传递

- **已在一生一芯等项目中得到应用，欢迎大家试用**

特性	DiffTest支持情况
场景	RISC-V, Cache, …
语言支持	Chisel, Verilog
多核支持	Yes
指令集模拟器(REF)	NEMU, Spike
仿真器	Verilator, VCS
调试插件	LightSSS, ChiselDB等

谢谢！
请批评指正