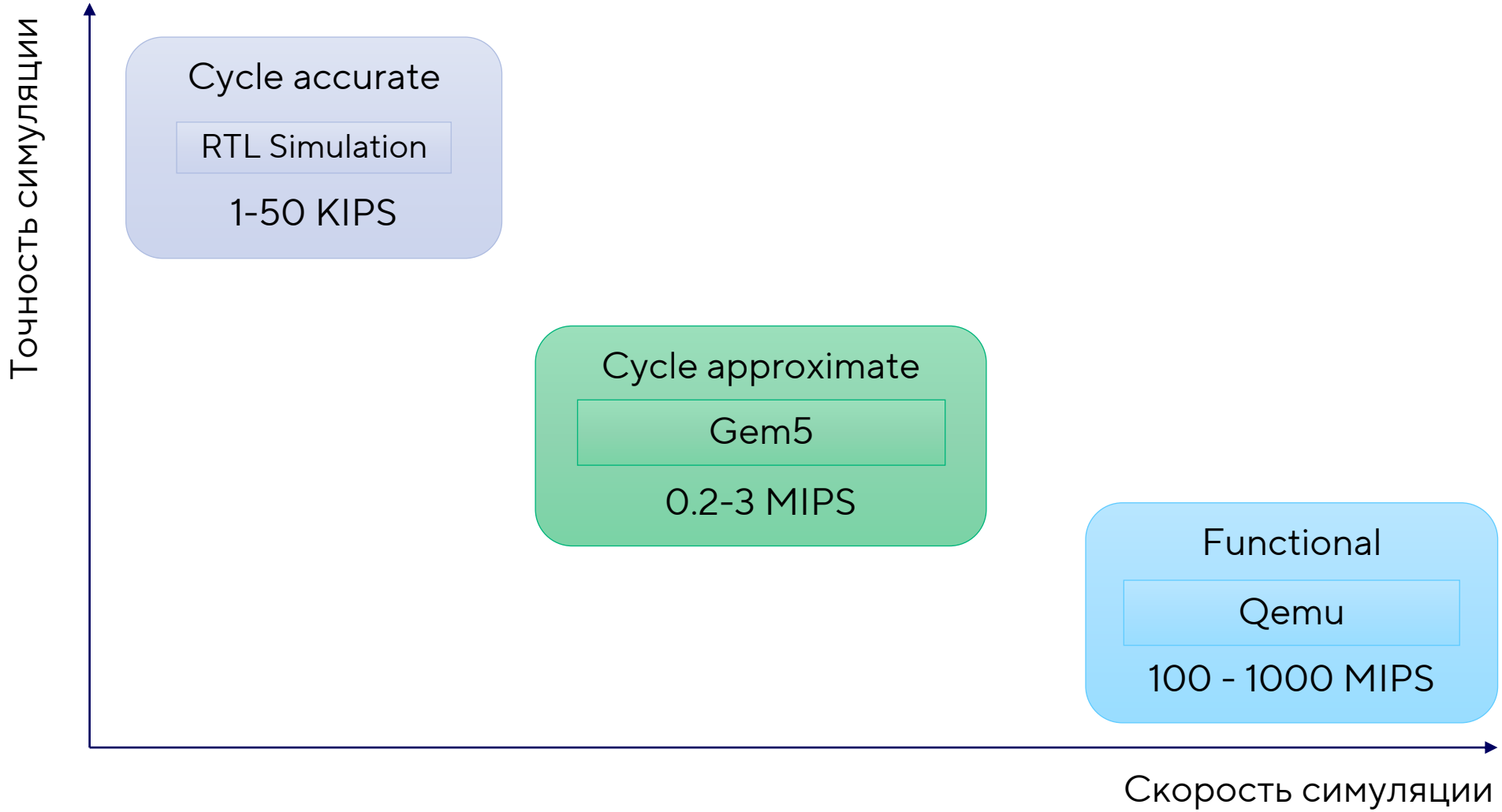


От Upstream до Silicon: как мы адаптировали Gem5 для разработки RISC-V CPU

Петров Валентин

VIII митап Альянса RISC-V,
Нижний Новгород, 14.11.2026

Что такое Performance-симулятор?



Для чего нужен Performance-симулятор

Анализ производительности железа — HW/SW co-design: новая фича сначала прототипируется в симуляторе и лишь затем реализуется в железе.

Анализ производительности софта на ещё не существующей платформе

Верификация – поиск performance-багов

Почему Gem5

Модели и режимы

- Event-driven simulation framework
- ISA: RISCV и др
- Baremetal/FullSystem/SyscallEmul
- CPU model switching (timing/Atomic, O3/Simple)

Аналитика

- Детальные статистики симуляции (perf counters)
- Трасса пайплайна
- Логирование

Экосистема

- Модульность
- Активное сообщество на Github

Кажется, что все готово «из коробки»

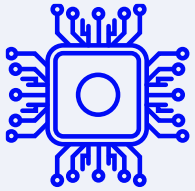
Как оценить сходимость модели?



Бенчмарки



HPM



Платформа



ROI



Pipeview



Бенчмарки: структура и устройство

Baremetal, no MMU

- DirectTests
- BPU only
- INT ALU, FP ALU, etc – lat/thpt
- Memory: L1->L2->L3->DDR; lat/bw
- Синтетические бенчмарки
- Реплики реальных приложений (SPEC, customer SW)

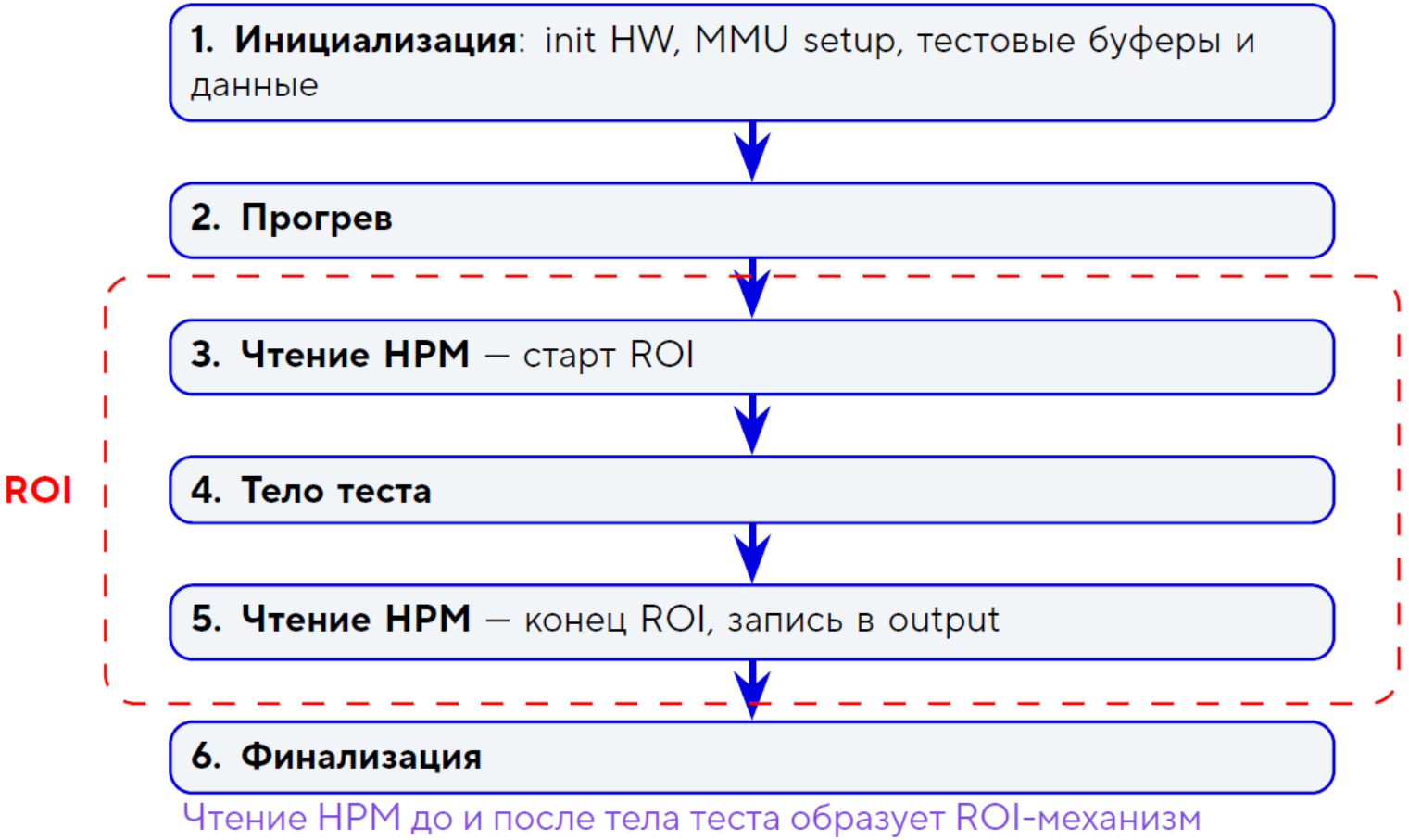
Baremetal + MMU

- Memory
- Реплики

Fullsystem (Linux)

- Реальные приложения

HW Performance monitors (HPM)



- CSR mhpmevent* - RISC-V interface

- BPU: call/ret/br hit/miss
- CPU: cyc, inst, stalls
- Cache: l1d/l2/l3 hit/mis/refill/evict
- Prefetcher, MMU, ..

Convergence Score

Бенчмарк	FPGA		gem5		DIFF (gem5/FPGA)	
	br_miss	ret_miss	br_miss	ret_miss	br_miss	ret_miss
foo	235	24	240	22	1.02	0.92
bar	25600	1380	25400	1340	0.99	0.97
geomean	—	—	—	—	**1.005**	**0.94**

BPU SCORE

0.975 – среднее по счётчикам (1.005 и 0.94)

Метрики

BPU | GEN_CYC | MMU | L1D | L2 | L3

Prefetch, NOC, DDR (план)

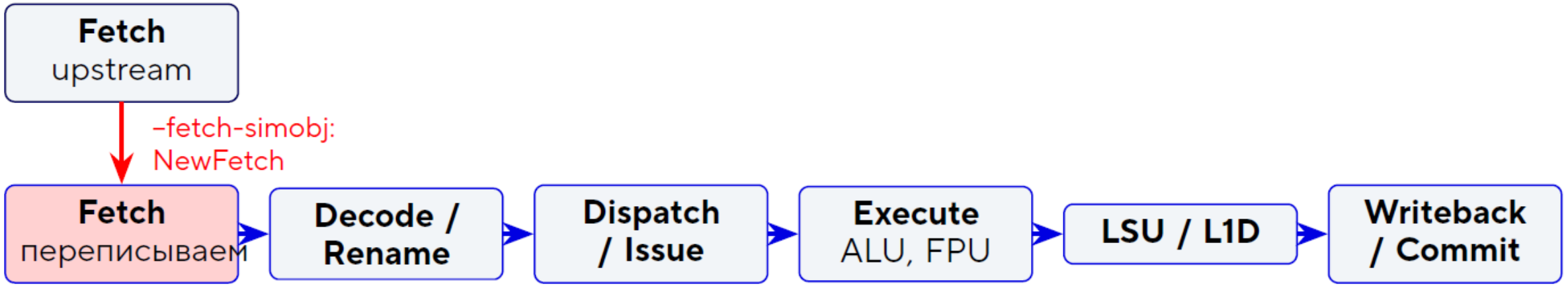
Name		Current
bpu_	yprof	0.937
gen_	yprof	0.895
l1d_pf_	yprof	0.000
l1d_	yprof	0.791
l1i_	yprof	0.620
l2_	yprof	0.126
l3_	yprof	0.051
mmu_	yprof	0.672

Уточнение модели

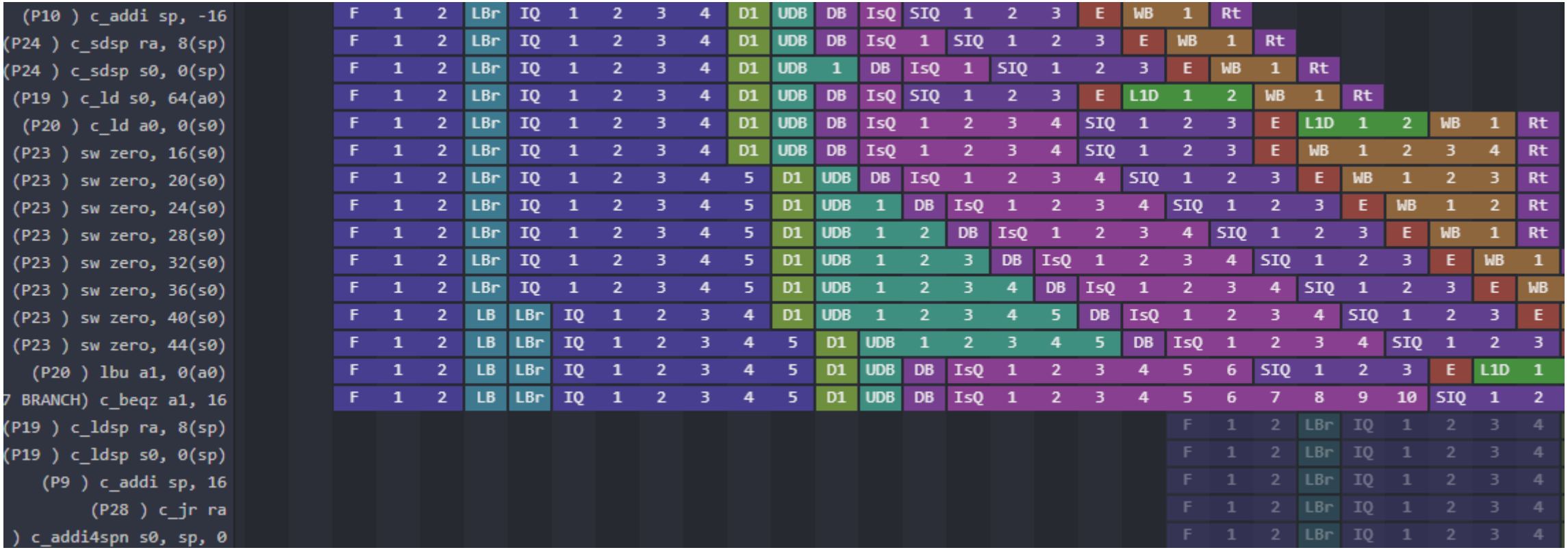
Видим низкий BPU Conv score

Насколько модель способна аппроксимировать железо

Нужен источник реальных данных об аппаратуре: MAS + RTLDev



Трассировка пайплайна



- Распространённый инструмент
- Open-source формат Konata
- Gem5 поддерживает «из коробки» - O3PipeView

Трассировка пайплайна

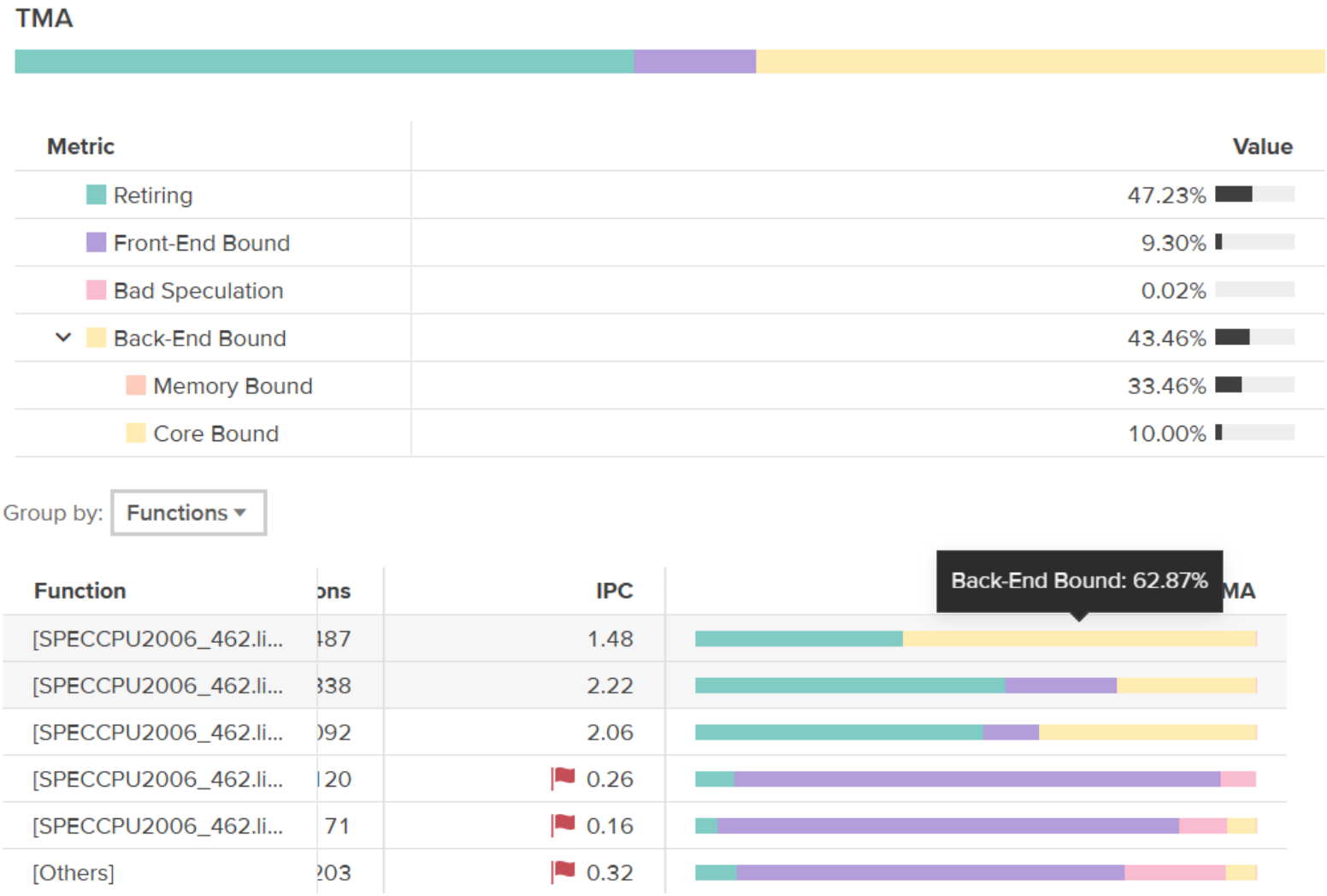
Минусы готовых решений

- Текстовый формат: гигантские трассы
- Медленная визуализация
- Ограниченная мета-информация
- Существенное замедление скорости симуляции

Решение

- Кастомный бинарный формат – **сжатие 100x**
- Ядро логгера в gem5 (буферизация, обработка в отдельном треде) – **высокая скорость**
- Стадии трассы **выровнены с RTL**
- Сравнение трасс **Gem5 vs RTL side-by-side**

TMA – скачок эффективности анализа

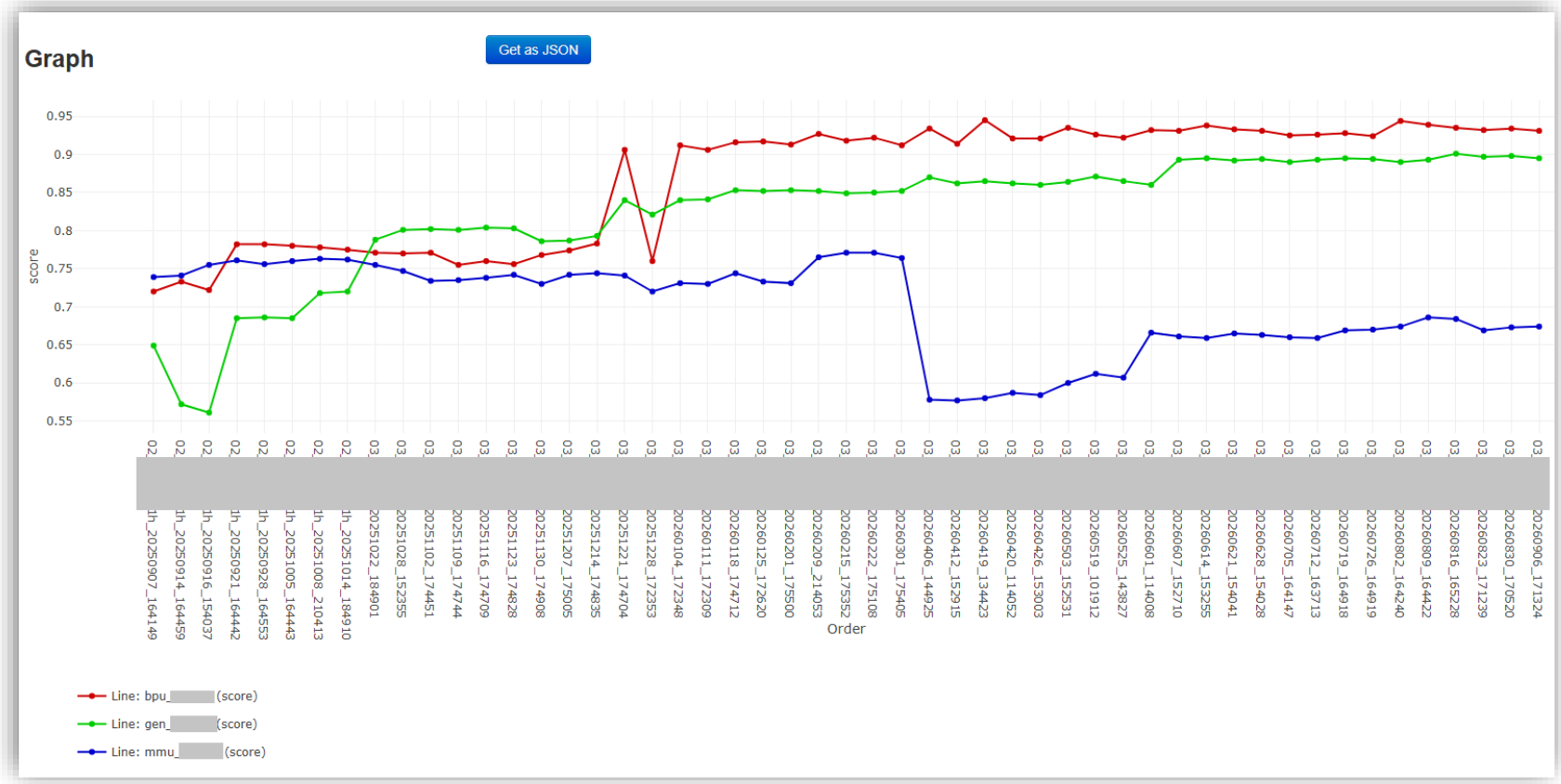


Автоматическое построение TMA по бинарной трассе

Vtune-like experience

Эффективный поиск узких мест

Все компоненты вместе



- **Автоматический циклический процесс** планомерного уточнения модели.
- **Викли регрессия**: сломали модель или поменялся RTL.
- Не автоматизировано самое сложное: **поиск причины расхождения** и написание модели.

Внедрение в цикл разработки

Репозиторий трасс

- QEMU + SPEC (app) -> трасса
- Выбор интервалов (art)
- Воссоздание Baremetal + MMU бинарника

Пример SPECINT06

- ~150 elfs, 50M инструкций
- Запуск на Gem5 в параллель – десятки минут
- Реконструируем оригинальный скор

Группа SW-HW codesign – ScoreHunters

- RTL_Dev, Benchmarks, Gem5, Analyzers, Tracing
- Весь стэк: параллельная симуляция, BinaryKonata -> TMA
- Реальные боттлнеки -> решения -> прототипы на Gem5
- Викли ревью с RTL_Dev

Пример сходимости с RTL

bench	RTL vs Model
sp_400.perlbench	-11.09%
sp_401.bzip2	-1.01%
sp_403.gcc	-13.15%
sp_429.mcf	-0.84%
sp_445.gobmk	4.98%
sp_456.hmmer	7.73%
sp_458.sjeng	0.50%
sp_462.libquantum	11.07%
sp_464.h264ref	-5.75%
sp_471.omnetpp	-9.25%
sp_473.astar	3.10%
sp_483.xalancbmk	19.77%
geomean	**0.09%**

Есть полностью выровненные паттерны

Случаются регрессии или изменения в аппаратуре

Целевая точность – около 10%

Заключение

На 4-й год проект Gem5 — в официальном пайплайне разработки новых фич производительности для CPU-кластера.

Сначала модель: оценка эффекта, только потом в разработку

Форк апстрима + ~**200K строк кода** (мерж с апстримом — боль).

Несколько поколений ядер, платформ, конфигураций кластера

Автоматический convergence: оценка сходимости и детект регрессий (и в модели, и в железе).

Большая кросс-командная работа:

- оптимизация софта
- оптимизация железа
- верификация
- развитие тулинга и инструментов анализа

Спасибо!

Будем на связи

БУДУЩЕЕ
В НАШИХ РУКАХ

Backup

	Доступность framework	Библиотека моделей, RISC-V reference	Лёгкость модификации моделей	Интерактивность, дебаггер, поддержка трасс, форматы вывода	Интеграция с внешними моделями
MARSS / MARSS++	★ Single Binary	★★★ Core, RAM, INTR	★★	★★★ Console, some logging	★★★★★ VirtIO, 9P
SPARTA	★★ Library	★★ Olympia Core	★	★★ Logging, debug harness	★★ SystemC models adapter
SystemC	★★★★★ Framework	★★ ISA and loosly-timed models	★★★	★★ Console, logging	★★★★★ Many integrations
Gem5	★★★ Framework	★★★★★ Many	★★	★★★★★★ Console, gdb, trace, logging, pipeview	★★★★★ SystemC, SST, RTL
SimEng	★★★ Framework	★★ Core only	★★★	★★ Console	★★ SST

Trade-off при написании модели

Максимально абстрактная модель

+ в Gem5 «из коробки»
+ высокая скорость симуляции
- невозможно достичь сходимости
- эксперименты без микроархитектурных деталей
- academia project

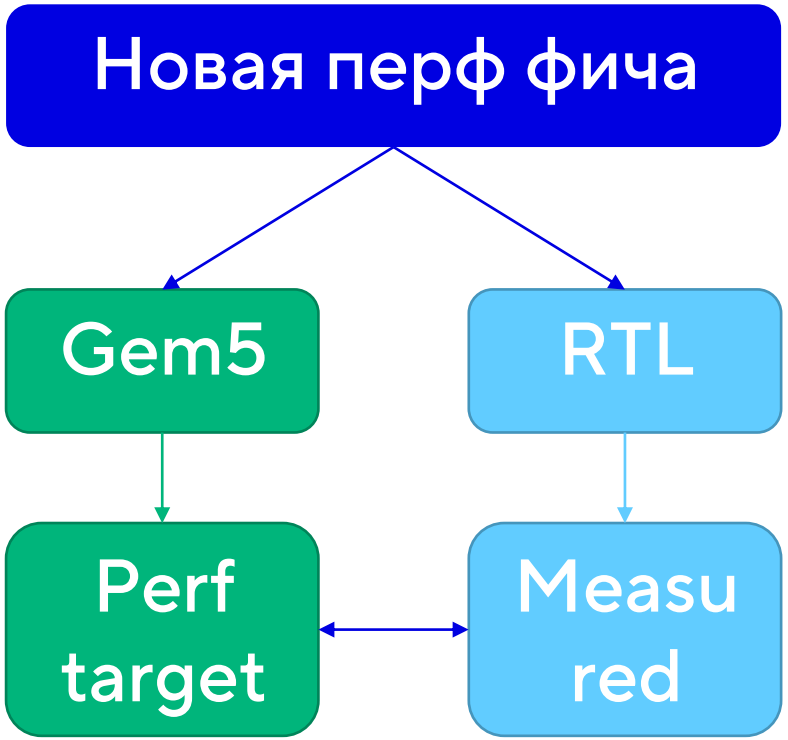
Точная RTL-модель

+ высокая точность/сходимость
+ эксперименты на уровне uArch
+ реальные HW HPM
- дорогая разработка
- сложность даже PoC
- медленная симуляция

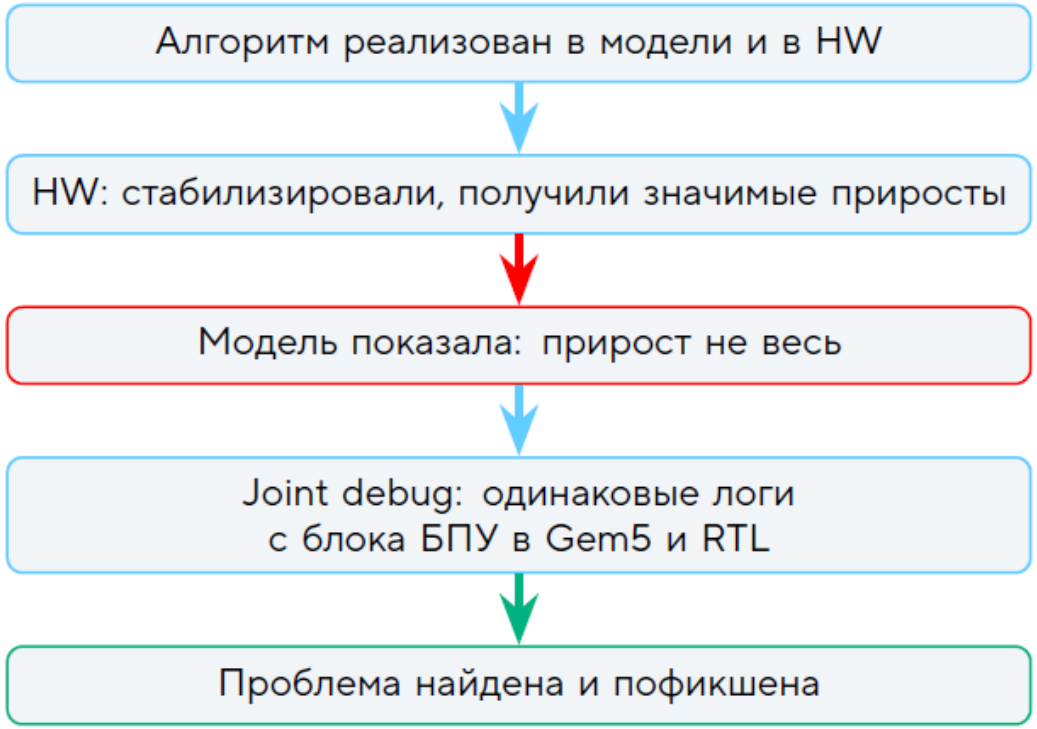
Баланс между точностью, репрезентативностью и абстракцией



Применение в верификации



Пример BPU:



Без перф-симулятора не было бы даже баг-репорта

Процесс внедрения

Зима 2022	Весна 2023	Лето 2023	Лето 2023	Осень 2023	Зима 2024	Весна 2024
• RTL_Dev работает без перф-симулятора	• точная модель фронтенда и БПУ	• первая версия точной микроарх. модели L1D/LSU	• существенно более точная MMU	• запрос команды фронтенда: смоделировать фичу БПУ	• точная модель dispatch/issue	• запросы на моделирование от L1D-команды RTL_Dev

все еще не production