РусКрипто XXVIII НАУЧНО-ПРАКТИЧЕСКАЯ КОНФЕРЕНЦИЯ

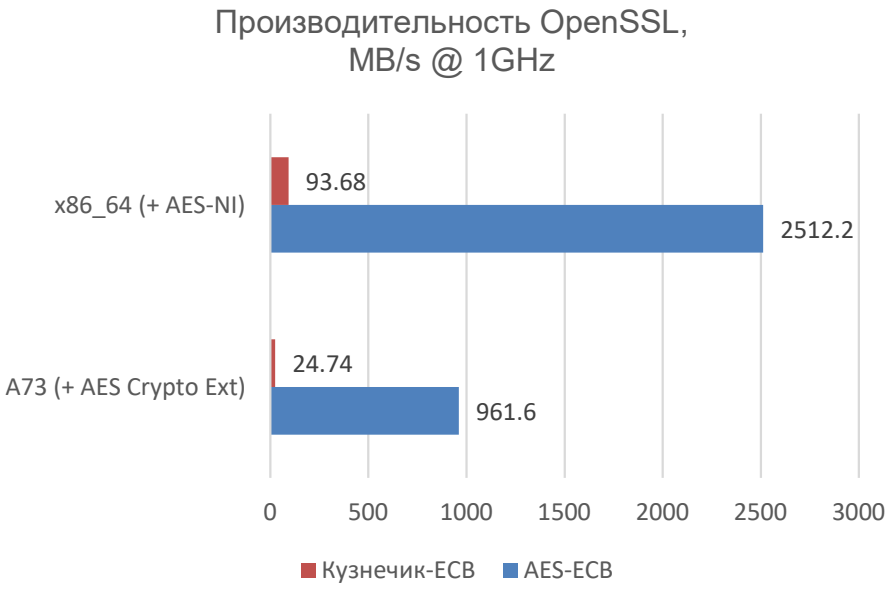
Векторное расширение RISC-V ISA для ГОСТ-криптографии

Матюков Андрей,
Альянс RISC-V

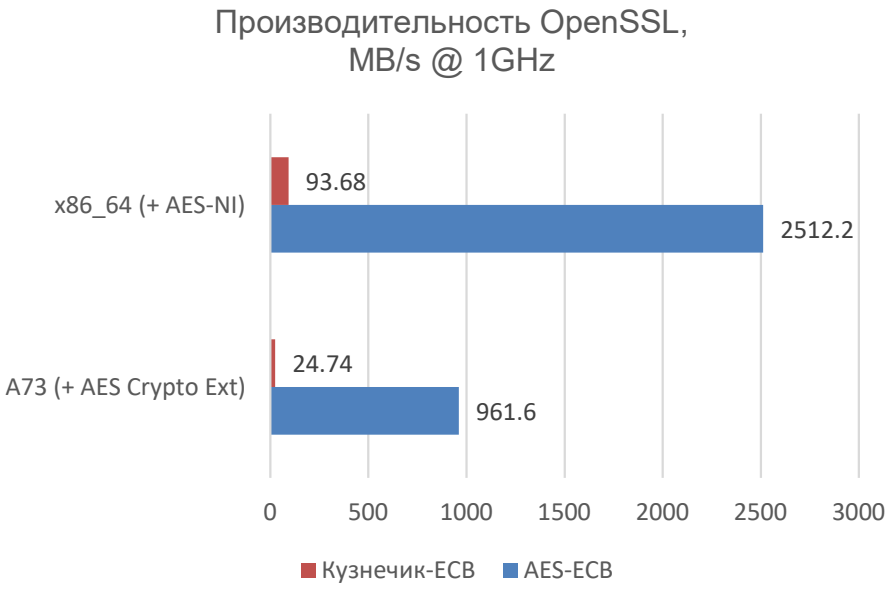
Требуемая пропускная способность в КФС

Низкая (1 – 10+ Кбит/с)	Средняя (1 – 100+ Мбит/с)	Высокая (100 Мбит/с – 1+ Гбит/с)
• Промышленные датчики t/P • Счётчики потребления энергии • Устройства «умного дома»	• АСУ ТП • Промышленные роботы с камерами машинного зрения • Умные станки • Умные светофоры	• Автономные автомобили (лидар + камеры + радары) • БПЛА • V2X-системы и умные города (дороги, транспортные потоки, камеры) • Цифровые заводы • 5G gNB

Проблема производительности



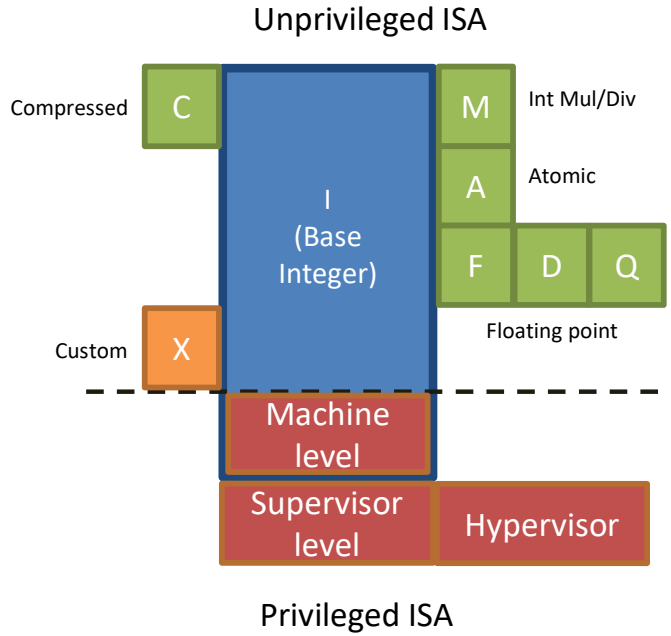
Проблема производительности



- AES
  - X86 AES-NI
  - ARM Crypto Extension
- Кузнечик
  - n/a

Почему RISC-V?

- Свободная и открытая процессорная архитектура
- Модульная и расширяемая система команд
- Подходит для широкого спектра оборудования (от smart-карт до HPC)
- Заложены механизмы создания пользовательских (custom) расширений



Расширения RISC-V ISA для ГОСТ-криптографии

Расширения RISC-V ISA для ГОСТ-криптографии

Scalar (Xkgost)

- Магма, Кузнечик, Стрибог
- Использует GPR-регистры
- Небольшой размер операндов (32-, 64-бит)
- Инструкции const-time
- Компактнее в SW
- Кратно улучшает производительность блочных шифров и криптографической хэш-функции
- Требует меньше площади
- Подходит для небольших ядер (embedded)

Расширения RISC-V ISA для ГОСТ-криптографии

Scalar (Xkgost)

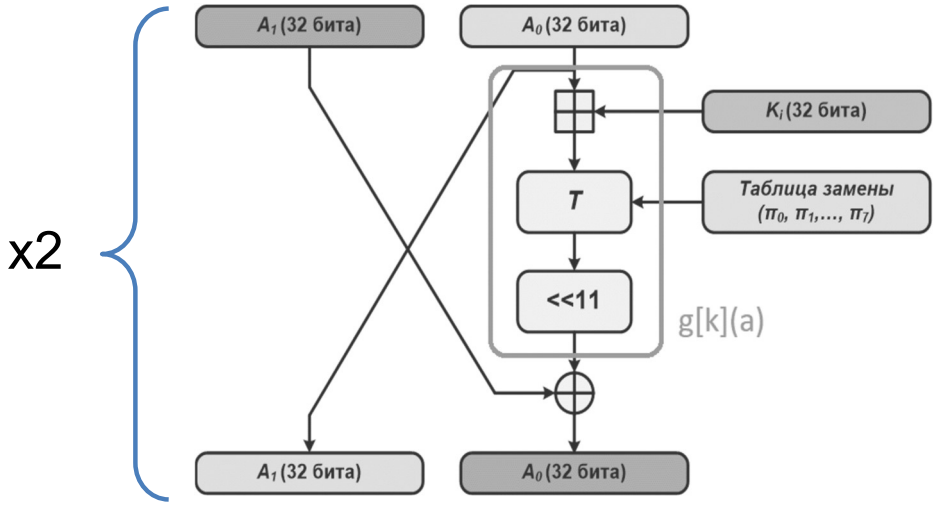
- Магма, Кузнечик, Стрибог
- Использует GPR-регистры
- Небольшой размер операндов (32-, 64-бит)
- Инструкции const-time
- Компактнее в SW
- Кратно улучшает производительность блочных шифров и криптографической хэш-функции
- Требует меньше площади
- Подходит для небольших ядер (embedded)

Vector (Xvkgost)

- Магма, Кузнечик, Стрибог
- Использует «широкие» векторные регистры
- Большой размер in/out операндов (128-, 512-бит)
- Еще более компактный SW код
- Еще более высокая производительность
- Инструкции const-time
- Выше «стоимость» реализации в железе (требует поддержки RVV)
- Подходит для старших ядер

Xvkgostm: Магма

Схема раунда Магма



vmagma2r.vv Vd, Vs2, Vs1
vmagma2r.vs Vd, Vs2, Vs1

Vd – выход (state)
Vs2 – вход (state)
Vs1 – раундовые ключи (пары)

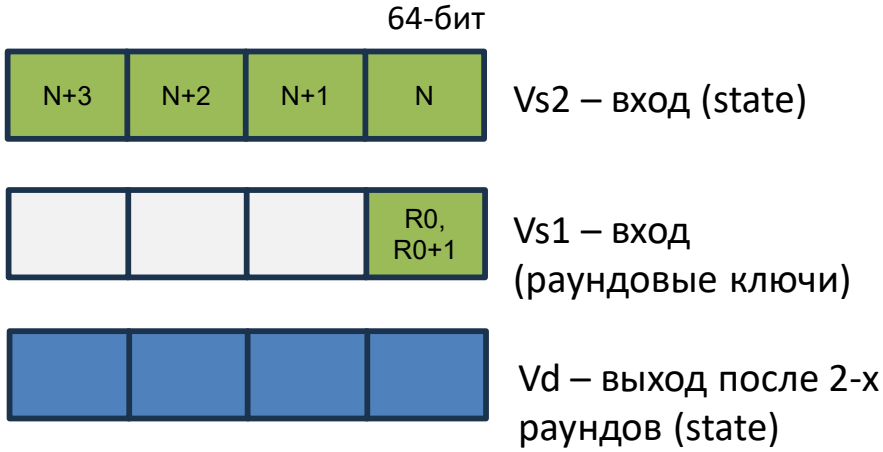
Вектор из групп элементов по 64-бита

** Все инструкции обладают свойством постоянного времени исполнения*

Xvkgostm: поддержка bulk и batch подходов

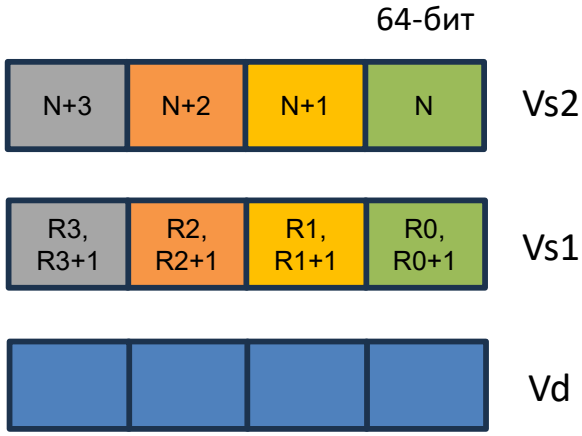
Bulk – один ключ на все группы элементов (блоки) вектора

vmagma2r.**vs**



Batch – для каждой группы элементов вектора свой ключ

vmagma2r.**vv**



Xvkgostm: Магма

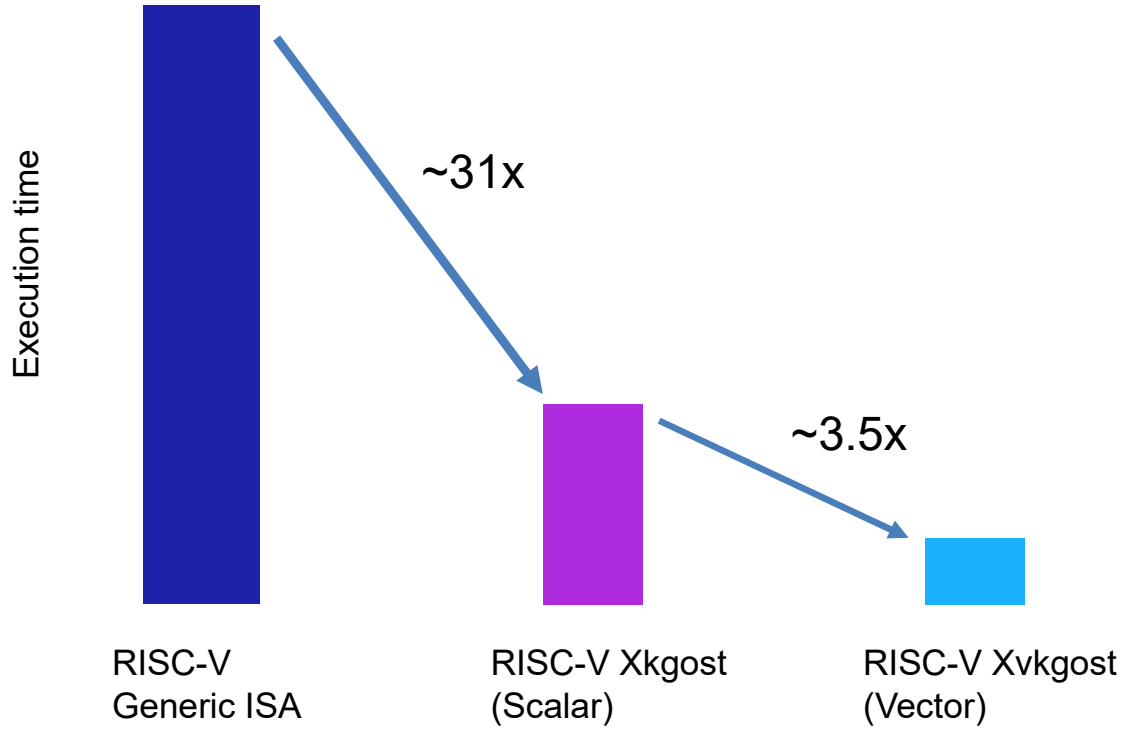
- 16 + 1 инструкций на блок
- Поддержка bulk и batch обработки
- Обработка до 16* независимых блоков при том же количестве инструкций

- Теоретическая производительность**:
  - 1 такт/байт ~ 7 Gbit/s @ 1GHz

```
for(; nblks > 0; nblks -= vl, in64 += vl, out64 += vl) {
    vl = __riscv_vsetvl_e64m8(nblks);

    /* load input blocks */
    vuint64m8_t state = __riscv_vle64_v_u64m8(in64, vl);

    /* 32 encryption rounds */
    state = __riscv_vmagma2r_vx_u64m8(state, rk1_0, vl);
    state = __riscv_vmagma2r_vx_u64m8(state, rk3_2, vl);
    state = __riscv_vmagma2r_vx_u64m8(state, rk5_4, vl);
    state = __riscv_vmagma2r_vx_u64m8(state, rk7_6, vl);

    state = __riscv_vmagma2r_vx_u64m8(state, rk1_0, vl);
    state = __riscv_vmagma2r_vx_u64m8(state, rk3_2, vl);
    state = __riscv_vmagma2r_vx_u64m8(state, rk5_4, vl);
    state = __riscv_vmagma2r_vx_u64m8(state, rk7_6, vl);

    state = __riscv_vmagma2r_vx_u64m8(state, rk1_0, vl);
    state = __riscv_vmagma2r_vx_u64m8(state, rk3_2, vl);
    state = __riscv_vmagma2r_vx_u64m8(state, rk5_4, vl);
    state = __riscv_vmagma2r_vx_u64m8(state, rk7_6, vl);

    state = __riscv_vmagma2r_vx_u64m8(state, rk6_7, vl);
    state = __riscv_vmagma2r_vx_u64m8(state, rk4_5, vl);
    state = __riscv_vmagma2r_vx_u64m8(state, rk2_3, vl);
    state = __riscv_vmagma2r_vx_u64m8(state, rk0_1, vl);

    /* swap 64-bit blocks (G*() compensation) */
    state = __riscv_vror_vx_u64m8(state, 32, vl);
    __riscv_vse64_v_u64m8(out64, state, vl);
}
```

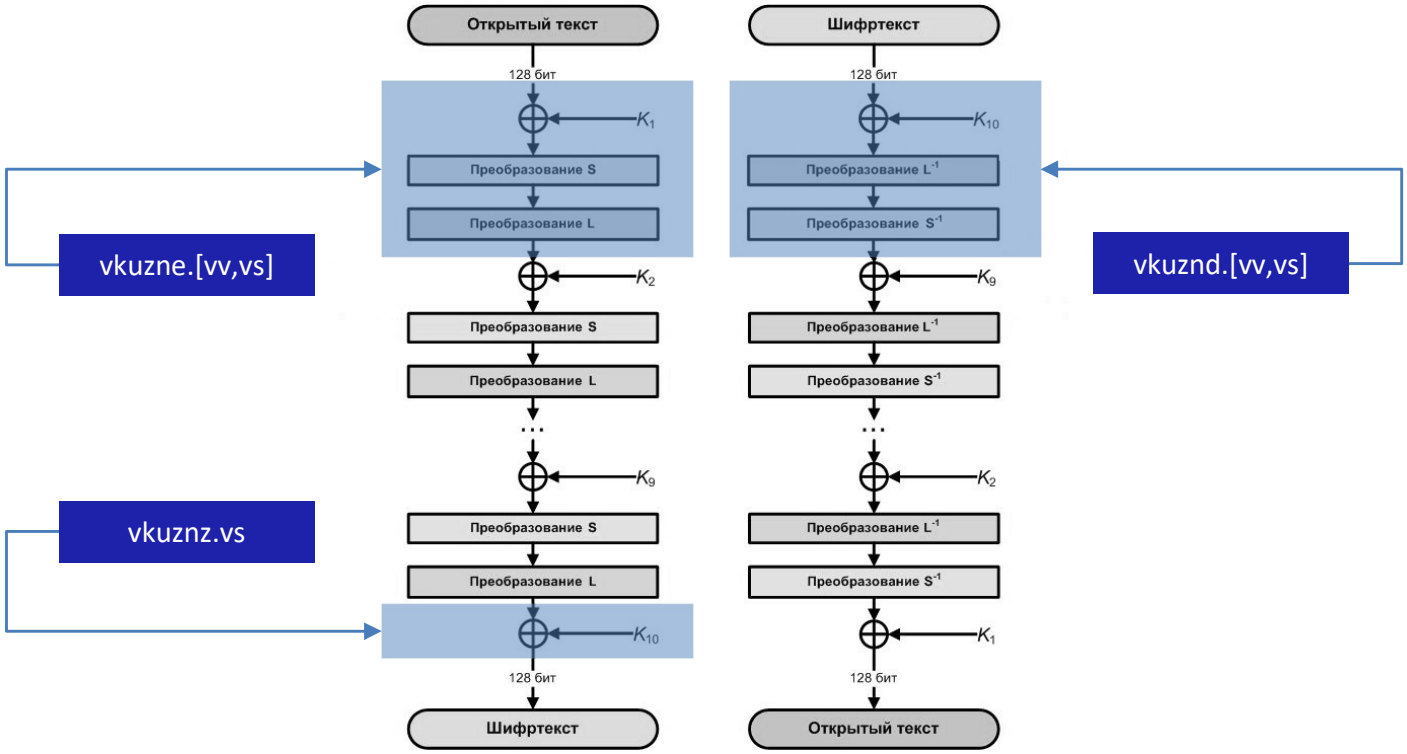
**VLEN=128*
***Режим ECB, VLEN=128, 1 вычислитель для Магмы, pipelined*
Может масштабироваться при увеличении VLEN и exec units

Ускорение Магма с RISC-V GOST ISA



**VLEN=128, 2 exec units, pipelined, прогноз*

Xvkgostk: Кузнечик



** Все инструкции обладают свойством постоянного времени исполнения*

vkuznz.vs Vd, Vs2, Vs1
vkuzne.[vv,vs] Vd, Vs2, Vs1
vkuznd.[vv,vs] Vd, Vs2, Vs1

Vd – выход (state)
Vs2 – вход (state)
Vs1 – раундовые ключи

Вектор из групп элементов по 128-бит

Xvkgostk: Кузнечик

- 9 + 1 инструкций на блок
- Поддержка bulk и batch обработки
- Обработка до 8* независимых блоков при том же количестве инструкций*

- Теоретическая производительность**:
  - 0.65 такта/байт ~ 11 Gbit/s @ 1GHz

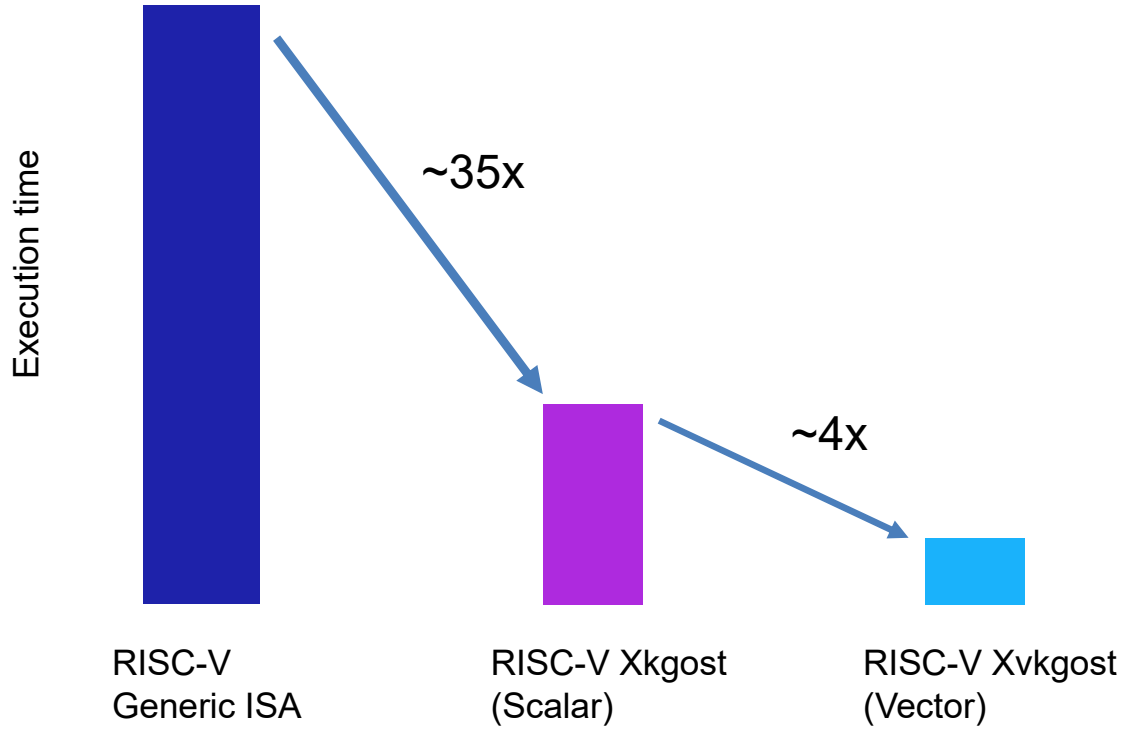
```
const uint64_t* in64 = (uint64_t*)in;
uint64_t* out64 = (uint64_t*)out;
size_t vl;

for (inl = NELEM(inl); inl > 0; inl -= vl, in64 += vl, out64 += vl) {
    vl = __riscv_vsetvl_e64m8(inl);

    vuint64m8_t state = __riscv_vle64_v_u64m8(in64, vl);

    state = __riscv_vkuzne_vs_u64m1_u64m8(state, rk0, vl);
    state = __riscv_vkuzne_vs_u64m1_u64m8(state, rk1, vl);
    state = __riscv_vkuzne_vs_u64m1_u64m8(state, rk2, vl);
    state = __riscv_vkuzne_vs_u64m1_u64m8(state, rk3, vl);
    state = __riscv_vkuzne_vs_u64m1_u64m8(state, rk4, vl);
    state = __riscv_vkuzne_vs_u64m1_u64m8(state, rk5, vl);
    state = __riscv_vkuzne_vs_u64m1_u64m8(state, rk6, vl);
    state = __riscv_vkuzne_vs_u64m1_u64m8(state, rk7, vl);
    state = __riscv_vkuzne_vs_u64m1_u64m8(state, rk8, vl);

    state = __riscv_vkuznz_vs_u64m1_u64m8(state, rk9, vl);

    __riscv_vse64_v_u64m8(out64, state, vl);
}
```

**VLEN=128*

***Режим ECB, VLEN=128, 1 вычислитель для Кузнечика, pipelined*

Может масштабироваться при увеличении VLEN и exec units

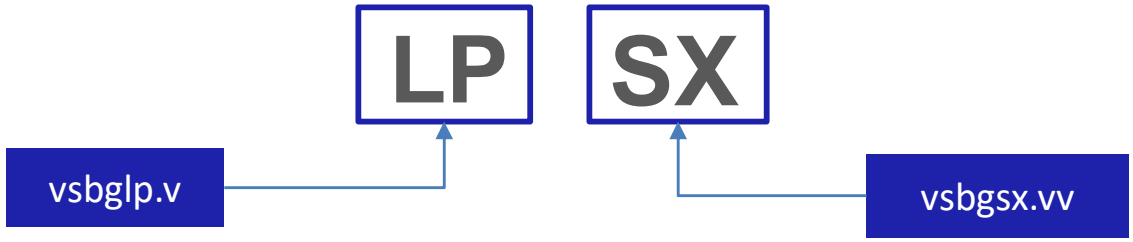
Ускорение Кузнечика с RISC-V GOST ISA



**VLEN=128, 2 exec units, pipelined, прогноз*

Xvkgosts: Стрибог

LPSX – основное преобразование функции сжатия $g\_N(h, m)$



vsbglp.v Vd, Vs2
vsbgsx.vv Vd, Vs2, Vs1

Вектор из групп элементов по 512-бит

Выводы

- Векторное расширение Xvkgost распространяет подход к ускорению отечественной криптографии на CPU на новые классы решений
- Два расширения перекрывают значительный диапазон областей применения в КФС (от IoT до высоконагруженных узлов)
- Позволяют использовать ГОСТ-алгоритмы в ранее недоступных низкоресурсных сегментах (IoT)
- Способствуют обеспечению свойств, необходимых в КФС:
  - Безопасность от timing side-channel
  - Энергоэффективность (меньше инструкций)
- Векторное расширение является альтернативой ускорителям по производительности при сохранении гибкости к адаптации SW реализаций под эволюционирующие требования к безопасности

Планы

- Ратификация Xvkgost в Альянсе RISC-V

- Новые направления работ:
  - Производительность перспективных алгоритмов ГОСТ PQC на RISC-V
  - Создание механизма использования защищенных ключей в алгоритмах шифрования ГОСТ на RISC-V

Спасибо
за внимание!

РусКрипто XXVIII НАУЧНО-ПРАКТИЧЕСКАЯ КОНФЕРЕНЦИЯ

Матюков Андрей
Альянс RISC-V

e-mail: a.matyukov@yadro.com
web: https://riscv-alliance.ru/