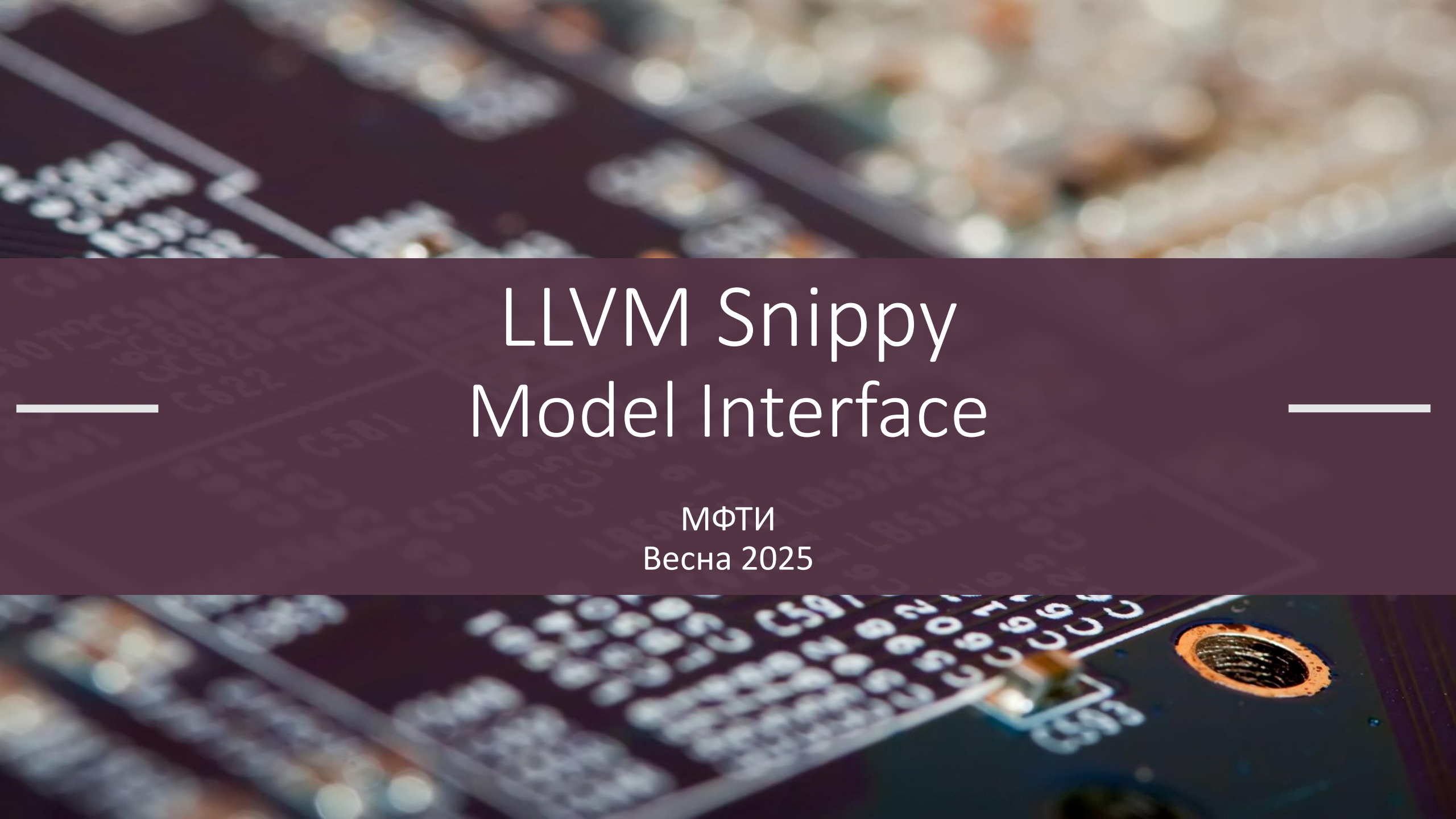


LLVM Snippy Model Interface

МФТИ
Весна 2025

Подключение модели к llvm-snippy

- Snippy имеет возможность напрямую подключать модель
- Зачем?

Подключение модели: сбор трассы

- Snippy умеет сразу же инициализировать модель и запускать сгенерированный сниппет для сбора трассы
- В общем случае трасса – набор отмеченных во времени событий
- Архитектурная трасса – упорядоченная последовательность выполненных инструкций с информацией об изменении регистров и обращениях в память
- Для простоты далее будем называть «архитектурную трассу» просто «трассой»

Подключение модели: сбор трассы

- Пример трассы от одного из RISC-V симуляторов:

```
addi X10, X0, 0x1
X10 <- 0x1
auipc X11, 0x1
X11 <- 0x1004
addi X11, X11, 0x20
X11 <- 0x1024
addi X12, X0, 0xf
X12 <- 0xf
addi X17, X0, 0x40
X17 <- 0x40
ecall
addi X10, X0, 0x0
X10 <- 0x0
addi X17, X0, 0x5d
X17 <- 0x5d
ecall
```

Отвлеченный пример: генерация float

- При генерации случайного сниппета с операциями с плавающей точкой есть вероятность возникновения в одном из регистров NaN значения
- Вскоре после первого регистра все остальные регистры будут заполнены NaN
- Создает ли это проблему для верификации?

Отвлеченный пример: генерация float

- При генерации случайного сниппета с операциями с плавающей точкой есть вероятность возникновения в одном из регистров NaN значения
- Вскоре после первого регистра все остальные регистры будут заполнены NaN
- Создает ли это проблему для верификации?
- Да, это ухудшает покрытие
- Можно ли это решить?

Отвлеченный пример: генерация float

- При генерации случайного сниппета с операциями с плавающей точкой есть вероятность возникновения в одном из регистров NaN значения
- Вскоре после первого регистра все остальные регистры будут заполнены NaN
- Создает ли это проблему для верификации?
- Да, это ухудшает покрытие
- Можно ли это решить?
  - Да, обновлять значения в регистрах
  - Значения можно обновлять раз в N инструкций
  - Значения можно обновлять как только накопилось N NaN’ов
    - Для это нам нужно знать содержимое регистров после каждой инструкции

Проблема: описание семантики инструкций

- Семантика – смысловое значение
- Семантика инструкций набора команд – смысловое значение инструкций и входящих в ее состав операндов
- В компиляторе в основном описывается трансляционная семантика
- Чтобы узнать как изменяется состояние регистров и памяти, нужно знание об интерпретационной семантике инструкций
- Где она уже описана?

Проблема: описание семантики инструкций

- Семантика – смысловое значение
- Семантика инструкций набора команд – смысловое значение инструкций и входящих в ее состав операндов
- В компиляторе в основном описывается трансляционная семантика
- Чтобы узнать как изменяется состояние регистров и памяти, нужно знание об интерпретационной семантике инструкций
- Где она уже описана?
  - **В симуляторе**

Подключение модели: использование для генерации

- Модель можно использовать как интерпретатор при генерации для получения состояния регистров и памяти
- Это можно использовать для разных режимов генерации:
  - FP overwrite
  - Самопроверка (selfcheck)
  - Косимуляция

Интерфейс модели

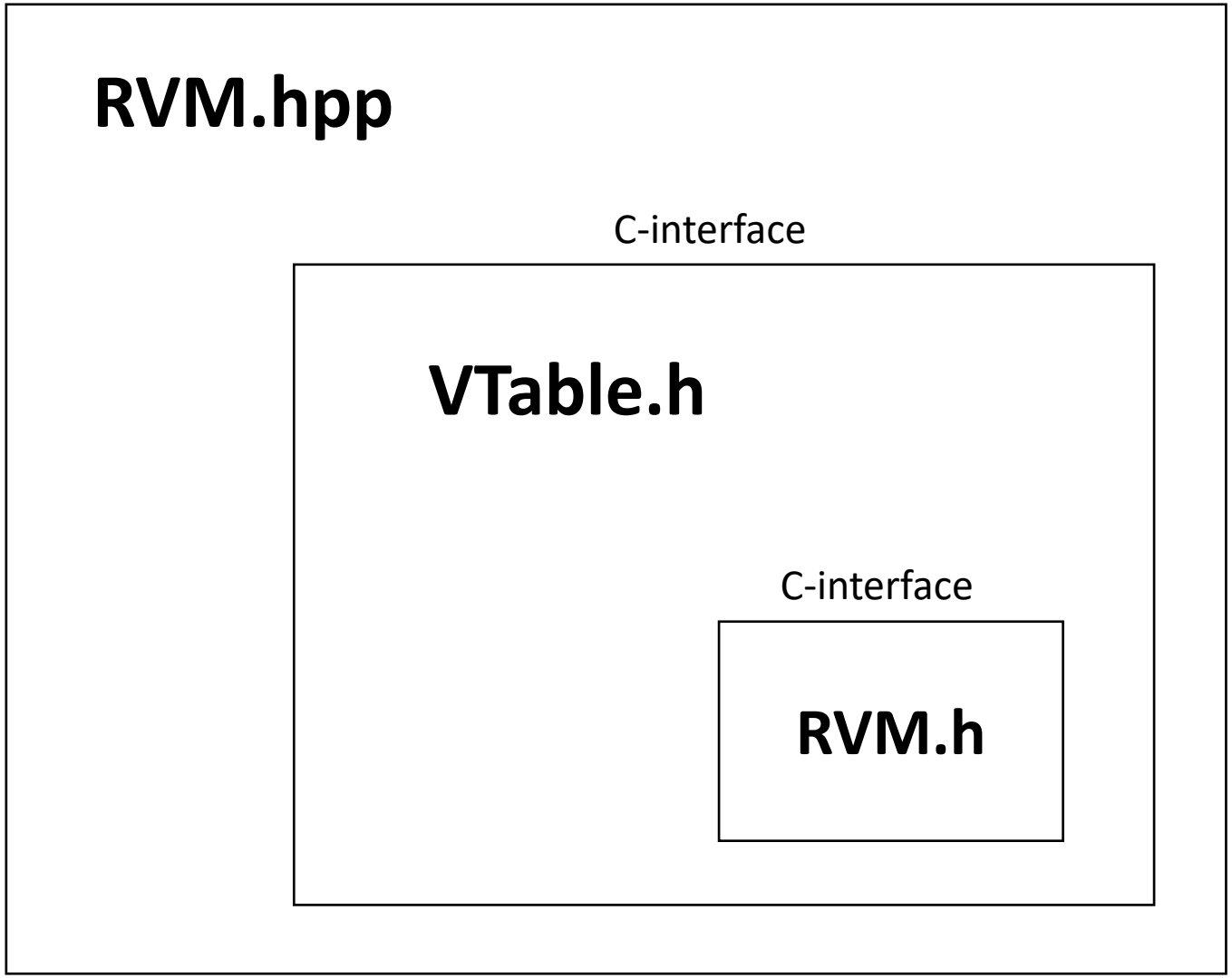
- Базовый интерфейс: что llvm-snippy может попросить от модели:

```
rvm_modelCreate                 rvm_readXReg    / rvm_setXReg
rvm_modelDestroy                rvm_readFReg    / rvm_setFReg
rvm_executeInstr                rvm_readVReg    / rvm_setVReg
rvm_readMem / rvm_writeMem      rvm_readCSRReg / rvm_setCSRReg
rvm_readPC  / rvm_setPC         rvm_logMessage
```

- Увы, необходимость поддерживать разные классы регистров делает модель бэкенд-специфичной. Для ARM или x86 интерфейс модели будет другим.

llvm/tools/llvm-snippy/Model/RISCVModel/RVM.h

StateType (RISCV Impl)



To be continued …

На следующем занятии

- Подробнее рассмотрим как происходит основная генерация в llvm-snippy