



# ІТМО

## Реализация модуля моделирования аппаратуры для SCADA-систем

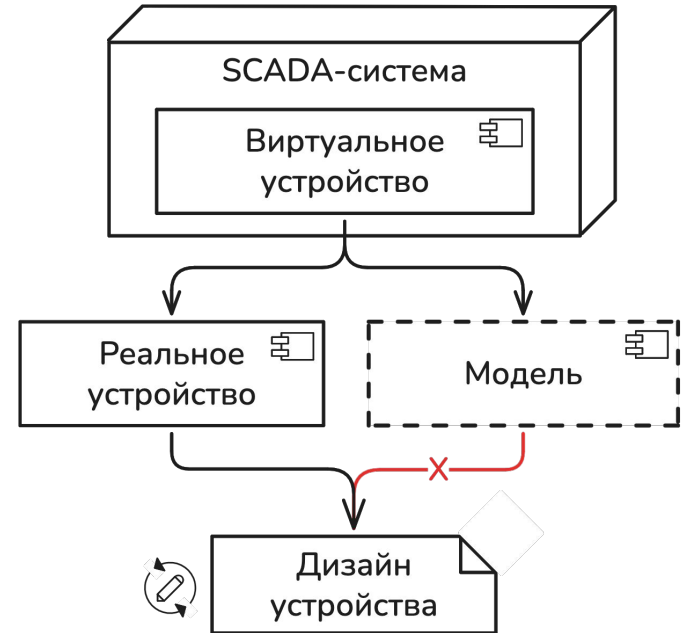
Студент: Тюрин Иван Николаевич, Р34102  
Руководитель: Осипов Святослав Владимирович  
Консультант: Нозик Александр Аркадьевич

# Предметная область

Для сквозного тестирования требуется полная система.

Виртуальные устройства снимают требование иметь реальные устройства при разработке системы.

Модель устройства может не существовать или отличаться от реального дизайна (HDL).



Использования виртуальных устройств  
в SCADA-системе

## Цель



Повышение качества систем на базе фреймворка «Controls.kt» за счёт их сквозного тестирования с применением моделей цифровых схем в виртуальных устройствах.

## Задачи

1. Исследование имеющихся решений для моделирования аппаратуры.
2. Разработка программного модуля для работы с моделями аппаратуры.
3. Демонстрация применения моделей в системе на базе Controls-kt.
4. Анализ полученных результатов.

# Фреймворк Controls.kt

ИТМО

## Назначение

- Разработка легковесных SCADA-систем;
- Моделирование программно-аппаратных систем.



## Особенности

- Асинхронная работа с устройствами;
- Виртуальные устройства;
- Язык программирования Kotlin.



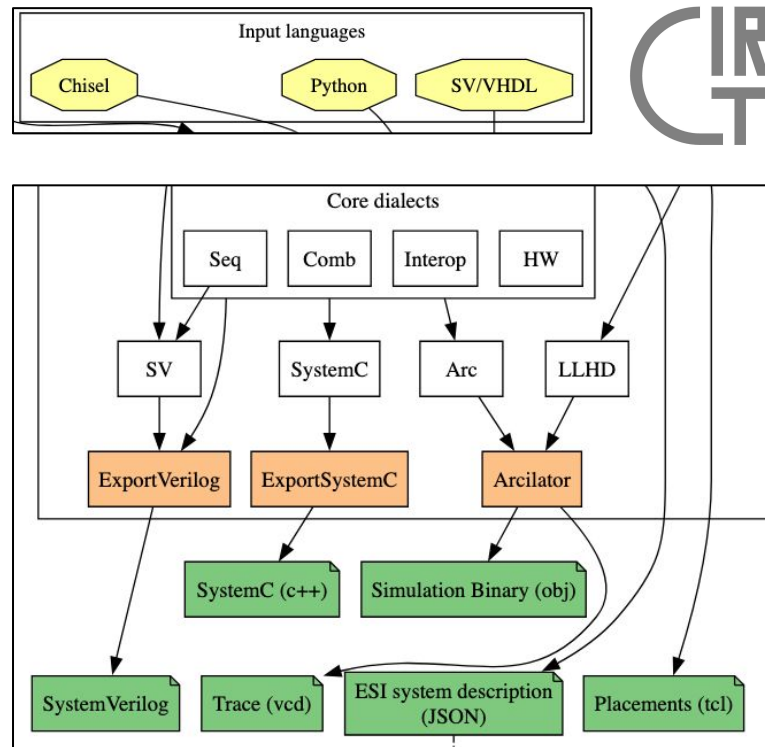
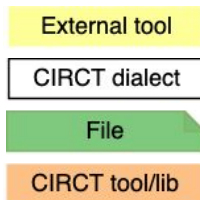
## Примеры использования

- Эксперимент по измерению массы нейтрино;
- Модель анализатора крови;
- Система управления отоплением жилого дома;
- Производственные линии и пр.

# Моделирование аппаратуры

## LLVM CIRCT – Circuit IR Compilers & Tools

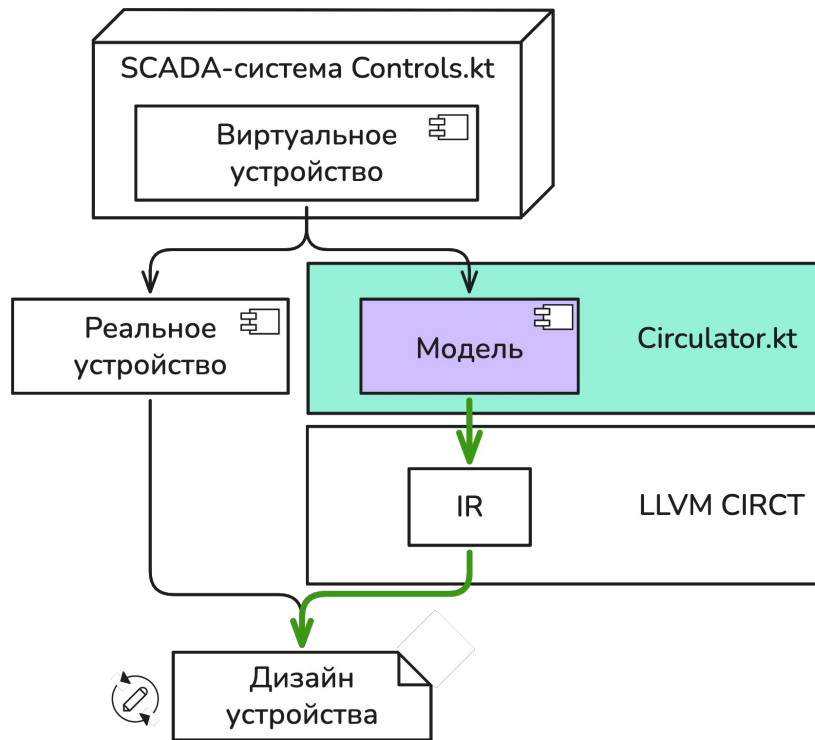
- Открытый развивающийся проект
- Инфраструктура LLVM
- Семейство диалектов MLIR для промежуточного представления
- «Language-agnostic» подход
- Симулятор цифровых схем **Arcilator**



# Разработанное решение

Проект Circulator.kt создает **явную** зависимость модели от HDL.

Промежуточное представление не ограничивает выбор языка описания аппаратуры.

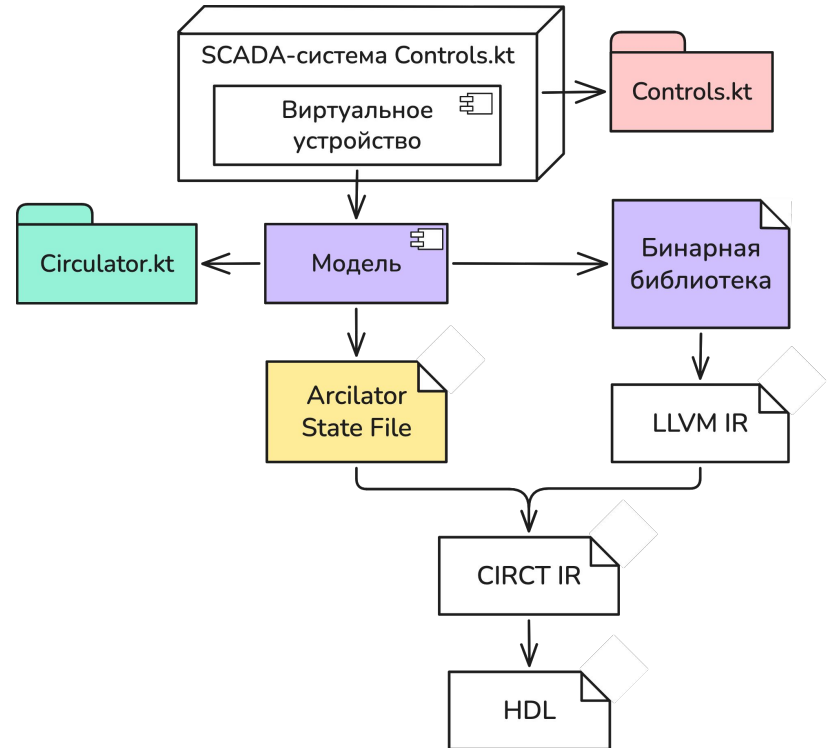


# Создание модели

Circulator.kt используется для создания программной модели.

Файл состояний Arcilator описывает интерфейс бинарной библиотеки.

Модель предоставляет API для вызова функций бинарной библиотеки.

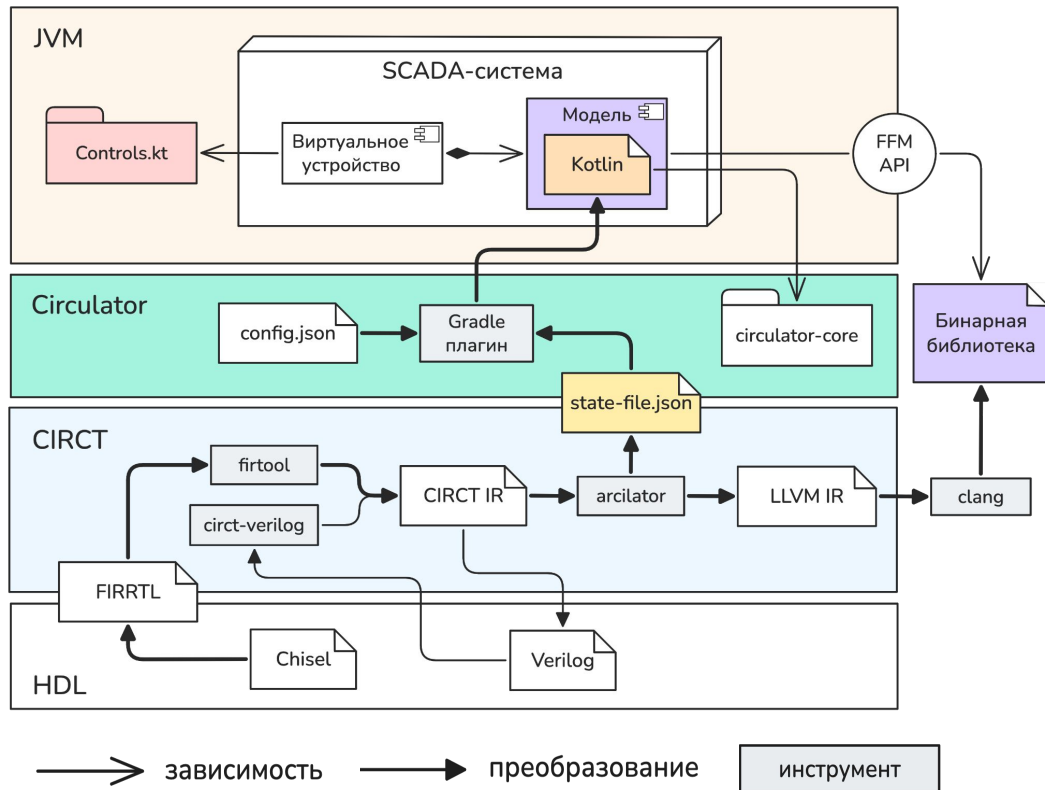


# Архитектура проекта

Circulator встраивается в компиляцию HDL.

Модуль **circulator-core** содержит абстракции для создания моделей.

**Gradle плагин** генерирует код моделей с учетом конфигурации.





# Схема данных Arcilator

## C++

```
/// Gathers information about a given Arc state.
struct StateInfo {
    enum Type { Input, Output, Register, Memory, Wire } type;
    std::string name;
    unsigned offset;
    unsigned numBits;
    unsigned memoryStride = 0; // byte separation between memory words
    unsigned memoryDepth = 0; // number of words in a memory
};
```

```
struct ModelInfo {
    std::string name;
    size_t numStateBytes;
    llvm::SmallVector<StateInfo> states;
    mlir::FlatSymbolRefAttr initialFnSym;
    mlir::FlatSymbolRefAttr finalFnSym;
```

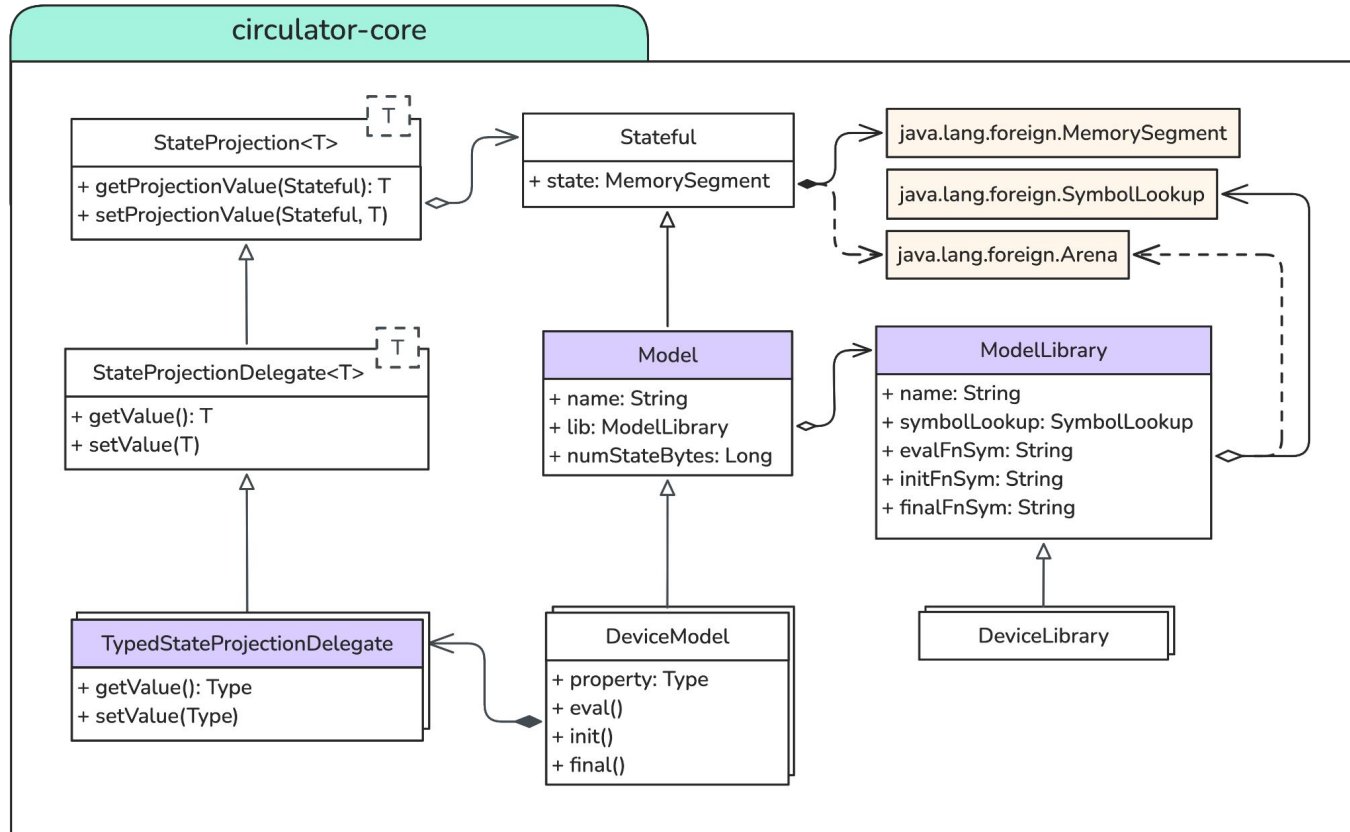
## Kotlin

```
public typealias StateFile = List<ModelInfo>
```

```
@Serializable
public data class StateInfo(
    val type: StateProjectionType,
    val name: String,
    val offset: UInt,
    val numBits: UInt,
    val memoryStrides: UInt = 0u, // bytes per word
    val memoryDepth: UInt = 0u, // number of words
)
```

```
@Serializable
public data class ModelInfo(
    val name: String,
    val numStateBytes: ULong,
    val initialFnSym: String,
    val finalFnSym: String,
    val states: List<StateInfo>
)
```

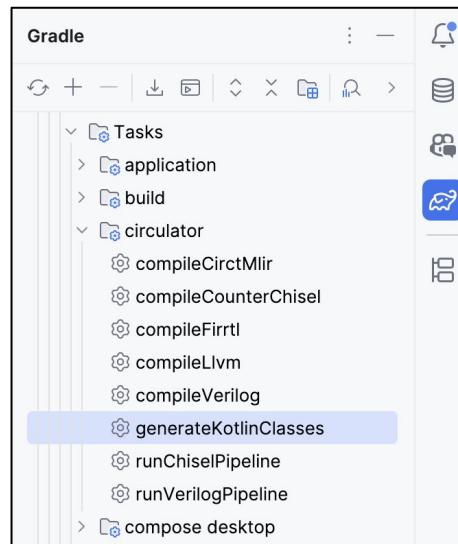
# Ключевые классы



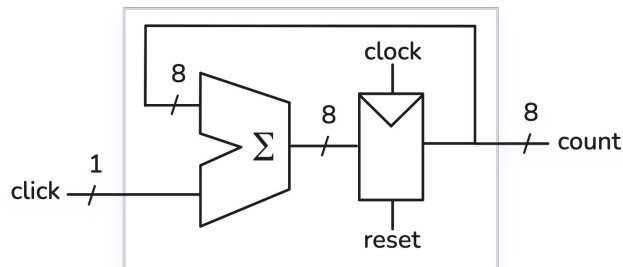
```
plugins {  
    /* ... */  
    id("io.github.e1turin.circulator.plugin")  
}  
  
circulator {  
    //config = file("config.json5")  
    config = PluginConfig(  
        mapOf(  
            "counter" to ModelConfig(  
                packageName = "io.github.e1turin.circulator.demo.generated",  
                stateFile = File("src/jvmTest/resources/arcilator/model-states.json"),  
                outputDirPath = "build/generated/sources/circulator/jvmMain/kotlin",  
                modelOptions = ModelOptions(  
                    open = true,  
                    allStatesOpen = true,  
                    allStatesMutable = true,  
                    allStatesType = StateType.entries,  
                    states = mapOf(  
                        "internalClk" to StateAttributes(  
                            open = true,  
                            mutable = true,  
                            access = true  
                        )  
                    )  
                ),  
            ),  
        libraryOptions = LibraryOptions(  
            open = true,  
        )  
    )  
})  
}
```

Конфигурация в файле  
`build.gradle.kts`

Плагин добавляет задачу для  
автоматической генерации кода



# Демонстрационная модель прибора



```
class ClickCounter extends Module {
  val io = IO(new Bundle {
    val click = Input(Bool())
    val count = Output(UInt(8.W))
  })

  val count = RegInit(0.U(8.W))
  io.count := count

  when(io.click) {
    count := count + 1.U
  }
}
```

Chisel

```
open class ClickCounterModel(
  state: MemorySegment,
  lib: ModelLibrary,
) : Model(MODEL_NAME, state, lib, NUM_STATE_BYTES) {
  open var clock: Byte by input<Byte>(0)

  open var reset: Byte by input<Byte>(1)

  open var io_click: Byte by input<Byte>(2)

  open val io_count: Byte by output<Byte>(11)

  fun eval() { /* ... */ }

  fun initial() { /* ... */ }

  fun `final`() { /* ... */ }

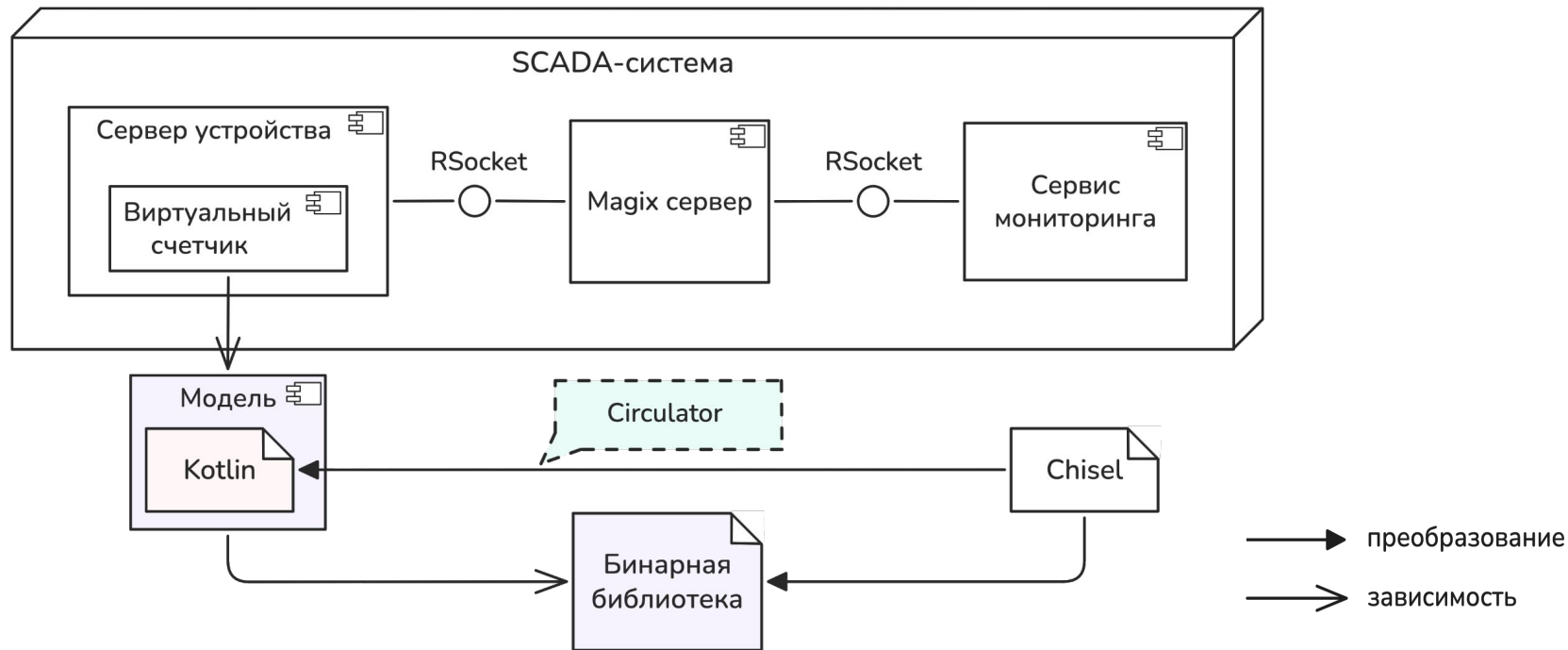
  companion object {
    const val MODEL_NAME: String = "ClickCounter"

    const val NUM_STATE_BYTES: Long = 12L

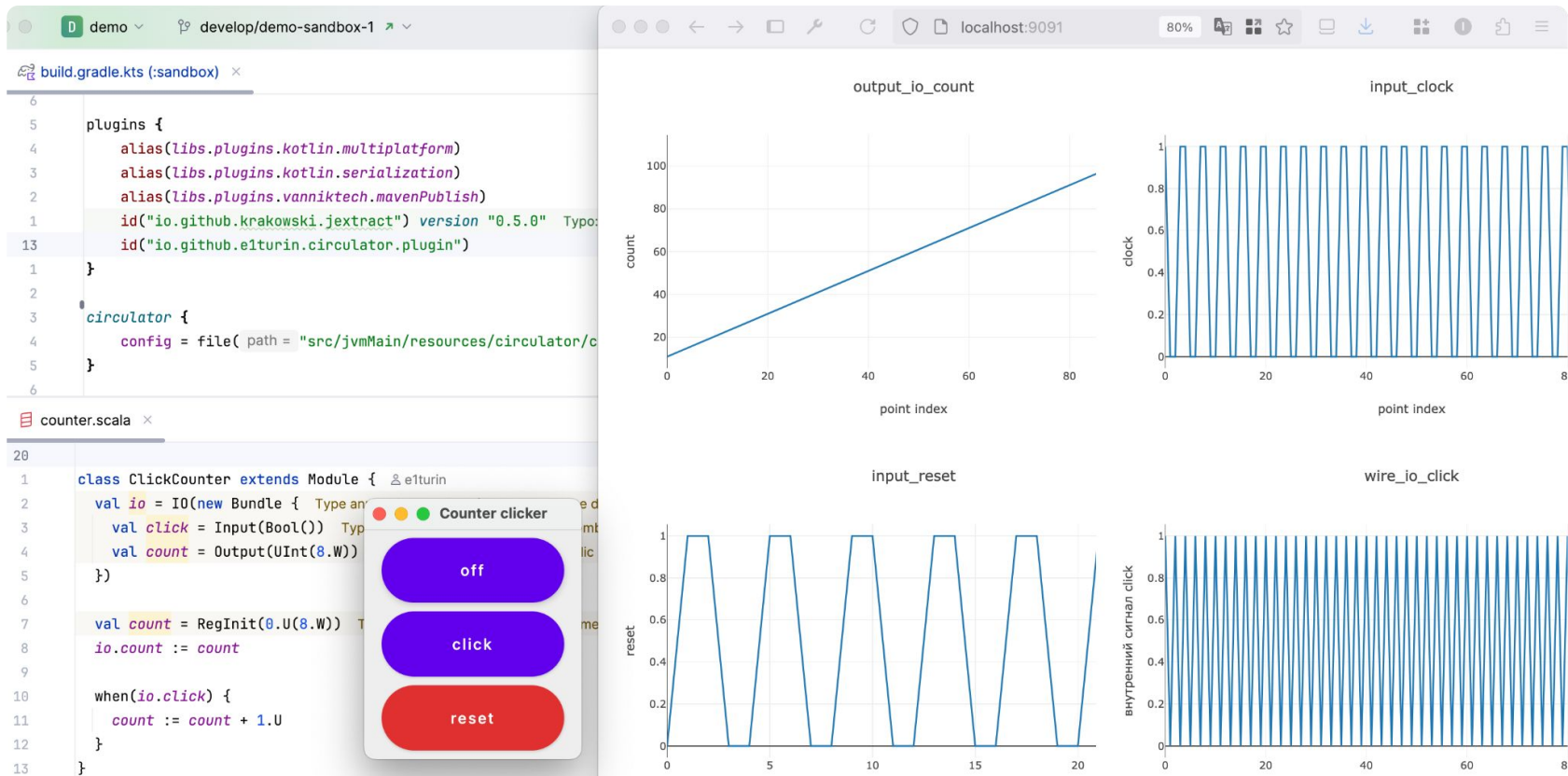
    fun instance(/* ... */): ClickCounterModel { /* ... */ }
  }
}
```

Kotlin

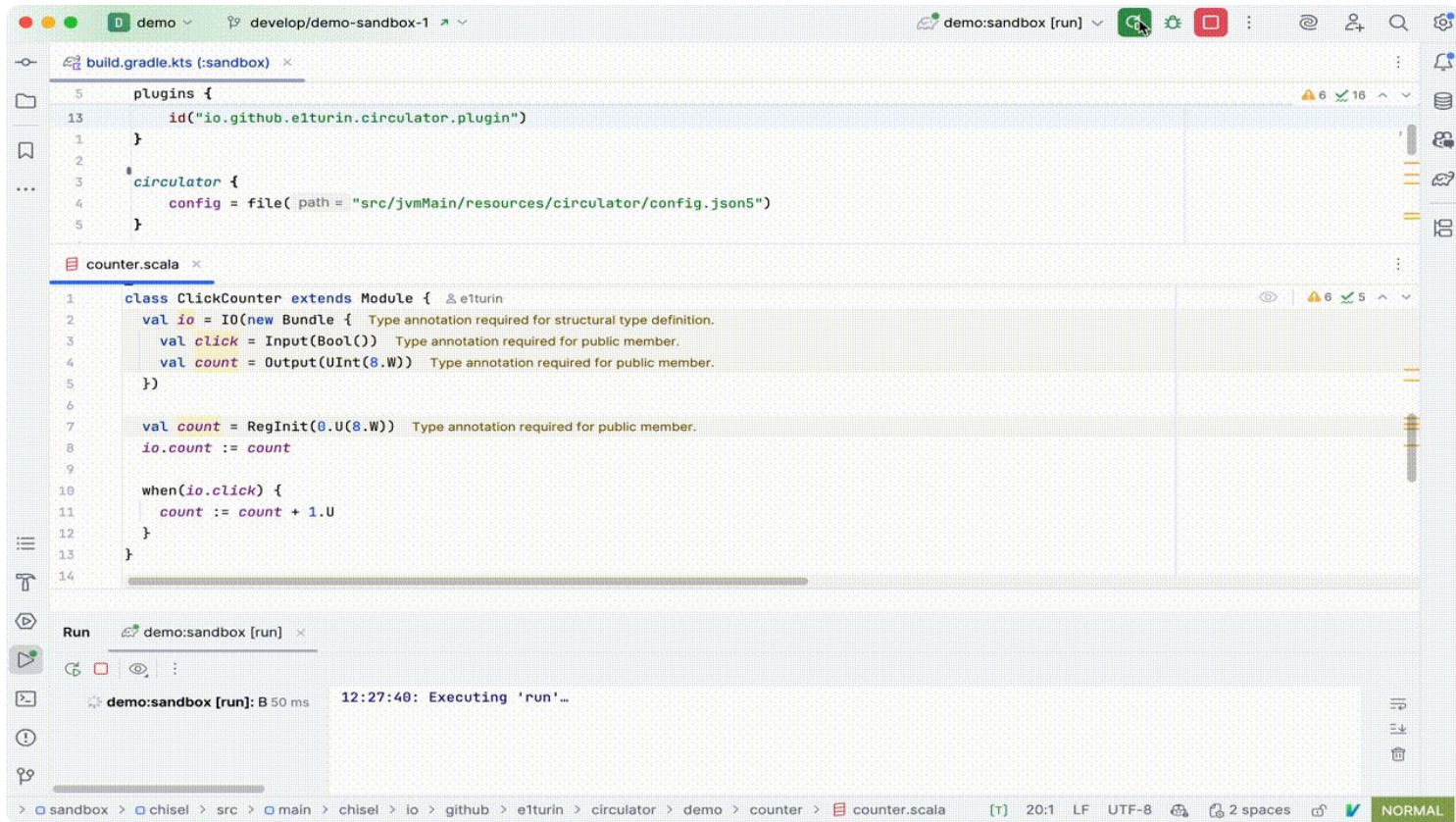
# Демонстрационная система



# Демонстрация работы



# Демонстрация работы



The screenshot shows an IDE window with two tabs: `build.gradle.kts (:sandbox)` and `counter.scala`. The `build.gradle.kts` tab is active, showing the following code:

```
5 plugins {  
13     id("io.github.e1turin.circulator.plugin")  
1  
2  
3     circulator {  
4         config = file(path = "src/jvmMain/resources/circulator/config.json5")  
5     }  
}
```

The `counter.scala` tab is also visible, showing the following code:

```
1 class ClickCounter extends Module { @ e1turin  
2     val io = IO(new Bundle { Type annotation required for structural type definition.  
3         val click = Input(Bool()) Type annotation required for public member.  
4         val count = Output(UInt(8.W)) Type annotation required for public member.  
5     })  
6  
7     val count = RegInit(0.U(8.W)) Type annotation required for public member.  
8     io.count := count  
9  
10    when(io.click) {  
11        count := count + 1.U  
12    }  
13  
14 }
```

At the bottom, the `Run` tab is active, showing the command `demo:sandbox [run]` and the output `demo:sandbox [run]: B 50 ms 12:27:40: Executing 'run'...`. The status bar at the bottom indicates the file path `> sandbox > chisel > src > main > chisel > io > github > e1turin > circulator > demo > counter > counter.scala`, the editor settings `[T] 20:1 LF UTF-8 2 spaces`, and the theme `NORMAL`.

# Результат работы

## Прежний подход

Требуется реальное устройство или функциональная модель для работы системы

Высокие трудозатраты на создание и обновление модели

Сквозное тестирование с моделями не затрагивает внутреннее устройство приборов

## Подход с Circulator



Реальное устройство в системе заменяется поведенческой моделью

Модель создается автоматически из HDL

При тестировании симулируется работа приборов на уровне цифровых схем



- Разработан подход для моделирования аппаратуры в «Controls.kt».
- Разработан программный модуль для создания моделей аппаратуры.
- Продемонстрирована возможность интеграции генерируемых моделей приборов с системой на базе «Controls.kt».
- Показано, как использование Circulator при тестировании позволяет повысить качество разрабатываемых систем.



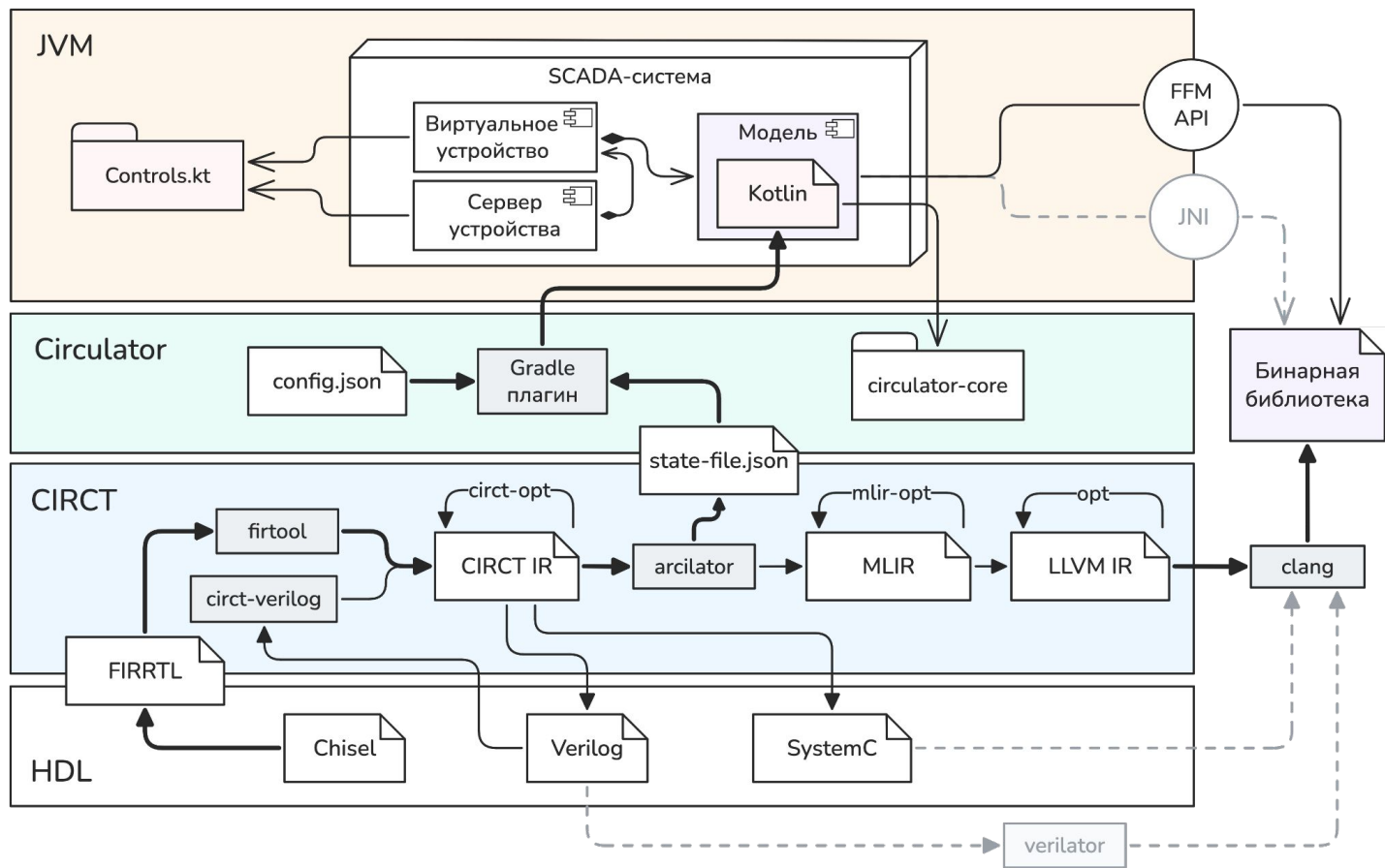
## Дальнейшее развитие

- Поддержка более старых версий JDK с помощью использования JNI.
- Поддержка других симуляторов HDL, например, Verilator.
- Поддержка платформ отличных от JVM, например, WASM.

**Спасибо  
за внимание!**

**it**'s **MO** *re than a*  
**UNIVERSITY**

[i1turin@yandex.ru](mailto:i1turin@yandex.ru)



инструмент

→ зависимость

→ преобразование

— реализовано

- - - потенциально возможно

# Ограничения Circulator

ІТМО



```

module {
  hw.module @ClickCounter(in %clock : !seq.clock, in %reset : i1, in
%io_click : i1, out io_count : i8) {
    %c1_i8 = hw.constant 1 : i8
    %c0_i8 = hw.constant 0 : i8
    %count = seq.firreg %1 clock %clock reset sync %reset, %c0_i8
    {firrtl.random_init_start = 0 : ui64, sv.namehint = "count"} : i8
    %0 = comb.add bin %count, %c1_i8 {sv.namehint = "_count_T"} : i8
    %1 = comb.mux bin %io_click, %0, %count : i8
    hw.output %count : i8
  }
  om.class @ClickCounter_Class(%basepath: !om.basepath) {
    om.class.fields
  }
}

```

```

[
  {
    "name": "ClickCounter",
    "numStateBytes": 12,
    "initialFnSym": "",
    "finalFnSym": "",
    "states": [
      {
        "name": "clock",
        "offset": 0,
        "numBits": 1,
        "type": "input"
      },
      {
        "name": "reset",
        "offset": 1,
        "numBits": 1,
        "type": "input"
      },
      {
        "name": "io_click",
        "offset": 2,
        "numBits": 1,
        "type": "input"
      },
      {
        "name": "clock",
        "offset": 3,
        "numBits": 1,
        "type": "wire"
      }, ... ]
    ]

```

## Прямое использование FFM API

- Сложный код
- Теряется семантика модели
- Гибкий подход

```
state[ValueLayout.JAVA_BYTE, 1] = 1 // reset <= 1
for (i in 1..7) {
    state[ValueLayout.JAVA_BYTE, 0] = 1 // clk <= 1
    dutEval.invokeExact(state)
    state[ValueLayout.JAVA_BYTE, 0] = 0 // clk <= 0
    dutEval.invokeExact(state)
}
state[ValueLayout.JAVA_BYTE, 1] = 0 // reset <= 0
```

## Jextract

- Сложный код
- Теряется семантика модели
- Требуется заголовочный файл Си

```
State.reset(state, 1)
for (i in 1..10) {
    State.clk(state, 1)
    counter_h.Counter_eval(state)
    State.clk(state, 0)
    counter_h.Counter_eval(state)
}
State.reset(state, 0)
```

```
// sizeof State == 8
struct State {
    char clock; // +0
    char reset; // +1
    /* ... */
    char _gap; // ???
    /* ... */
    char o; // +7
};

extern void
Device_eval(void *state);
```

## Circulator

- Легко читаемый код
- Без дополнительных артефактов
- Сохранена семантика модели
- Свободное расширение моделей

```
counter.reset = 1
for (i in 1..7) {
    counter.clock = 1
    counter.eval()
    counter.clock = 0
    counter.eval()
}
counter.reset = 0
```

```
counter.eval {
    rst = 1.bit
    counter.eval(10) { clk = !clk }
    rst = 1.bit
}
```

```

fun runRawFfmApiForCounter(n: Int = 10): Int {
    val lookup = SymbolLookup.libraryLookup(properLibName, Arena.ofConfined().use {
        .or(SymbolLookup.loaderLookup())
        .or(Linker.nativeLinker().defaultLookup())
    })

    val linker = Linker.nativeLinker()

    val dutEval = linker.downcallHandle(
        lookup.find("Counter_eval").get(),
        FunctionDescriptor.ofVoid(ValueLayout.ADDRESS)
    )

    val stateLayout = MemoryLayout.sequenceLayout(8, ValueLayout.ADDRESS)
    val arena = Arena.ofConfined().use {
        val state = it.allocate(8)

        state[ValueLayout.JAVA_BYTE, 1] = 1 // reset <= 1
        for (i in 1..7) {
            state[ValueLayout.JAVA_BYTE, 0] = 1 // clk <= 1
            dutEval.invokeExact(state)
            state[ValueLayout.JAVA_BYTE, 0] = 0 // clk <= 0
            dutEval.invokeExact(state)
        }
        state[ValueLayout.JAVA_BYTE, 1] = 0 // reset <= 0

        for (i in 1..n) {
            state[ValueLayout.JAVA_BYTE, 0] = 1 // clk <= 1
            dutEval.invokeExact(state)
            state[ValueLayout.JAVA_BYTE, 0] = 0 // clk <= 0
            dutEval.invokeExact(state)
        }

        return state[ValueLayout.JAVA_BYTE, 7].toInt()
    }
}

```

```

fun runJextractFfmApiForCounter(n: Int = 10): Int {
    val arena = Arena.ofConfined().use {
        val state = it.allocate(State.layout())

        State.reset(state, 1)
        for (i in 1..10) {
            State.clk(state, 1)
            counter_h.Counter_eval(state)
            State.clk(state, 0)
            counter_h.Counter_eval(state)
        }
        State.reset(state, 0)

        for (i in 1..n) {
            State.clk(state, 1)
            counter_h.Counter_eval(state)
            State.clk(state, 0)
            counter_h.Counter_eval(state)
        }

        return state[ValueLayout.JAVA_BYTE, 7].toInt()
    }
}

```

```

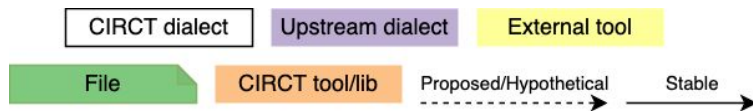
fun runCirculator(n: Int = 10): Int {
    val arena = Arena.ofConfined().use { arena ->
        val counter = CounterChiselModel.instance(arena)

        counter.reset = 1
        for (i in 1..7) {
            counter.clock = 1
            counter.eval()
            counter.clock = 0
            counter.eval()
        }
        counter.reset = 0

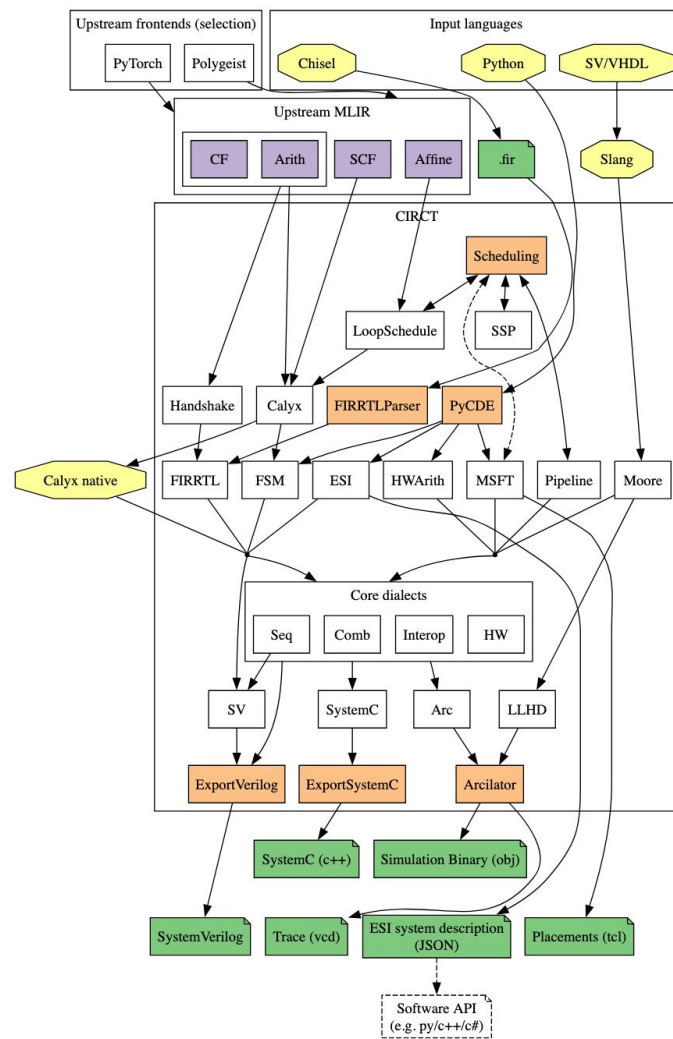
        for (i in 1..n) {
            counter.clock = 1
            counter.eval()
            counter.clock = 0
            counter.eval()
        }

        return counter.count.toInt()
    }
}

```



<https://circt.llvm.org/>





# Демонстрация работы

ІТМО

