Федеральное государственное бюджетное учреждение науки
Институт системного программирования РАН

Автоматизация тестирования моделей аппаратуры на основе статического анализа HDL-описаний

Смолов Сергей Александрович

Научный руководитель
д.ф.-м.н. Петренко Александр Константинович

Содержание

- Введение в предметную область
- Обзор существующих подходов
- Постановка задачи
- Идея решения
- Текущие результаты
- Перспективы

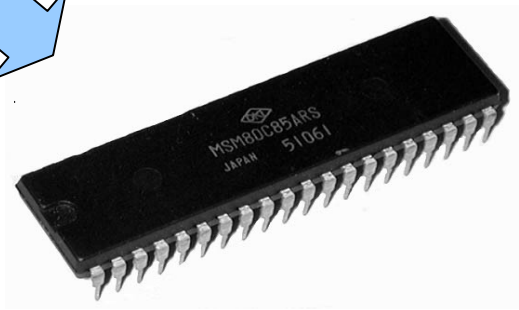
Упрощенная схема разработки цифровой аппаратуры

Требования

проектирование

HDL-описание

синтез



ИСП РАН

HDL (Hardware Definition Language)

- Много общего с ЯП (Pascal, C)
- Есть средства взаимодействия с ЯП (интерфейсы VPI, VHPI)
- Модули на HDL могут исполняться на симуляторах (Icarus Verilog, GHDL, …)
- **VHDL, Verilog**

Основные компоненты HDL-описаний

- Модуль
  - Интерфейс (i/o ports)
  - Реализация (body, state)
- Процесс
  - Параллельность
  - Список чувствительности (sensitivity list)
    - Сигналы синхронизации (clocks)

Некоторые аспекты тестирования аппаратуры

- Невозможность исправления готовых микросхем
  - Верификация HDL-описаний
- Рост сложности проектов
  - IP-ядра
- Трудоемкость разработки тестов вручную (J. Bergeron, 2006)
  - 70% от всего времени разработки
  - 80% от общего числа строк кода

Методы верификации: формальный подход

Характеристики:

- Математическое доказательство корректности
- Исчерпывающий (!!!)
- Область применения - поздние стадии проектирования

Примеры:

- Проверка эквивалентности
  - *Сравнение HDL с другими представлениями*
- Проверка свойств
  - *Формулы темпоральной логики*

Инструменты: PSL, UPPAAL ...

ИСП РАН

Методы верификации: имитационный подход

Характеристики:

- Запуск в симулируемом окружении
- Не исчерпывающий
- Возможно использование абстракций (!!!)

Примеры:

- Покрытие условий/кода/ветвей функциональности
- Тесты случайные/ограничения/направленные

Инструменты:

OS-VVM, SystemVerilog, ZamiaCAD, C++TesK HW Edition ...

Проблемы

- Имитационный:
  - Ориентация на один HDL-язык
  - Низкий уровень автоматизации при разработке сложных (не-случайных) тестов
- Формальный:
  - Разработка модели\формул — не автоматизирована

Постановка задачи

Предложить метод и разработать инструменты для автоматизации построения тестовых систем для верификации аппаратуры

- Независимость от HDL-языка
- Автоматическая генерация тестов (как случайных, так и более сложных)
  - Автоматные тесты

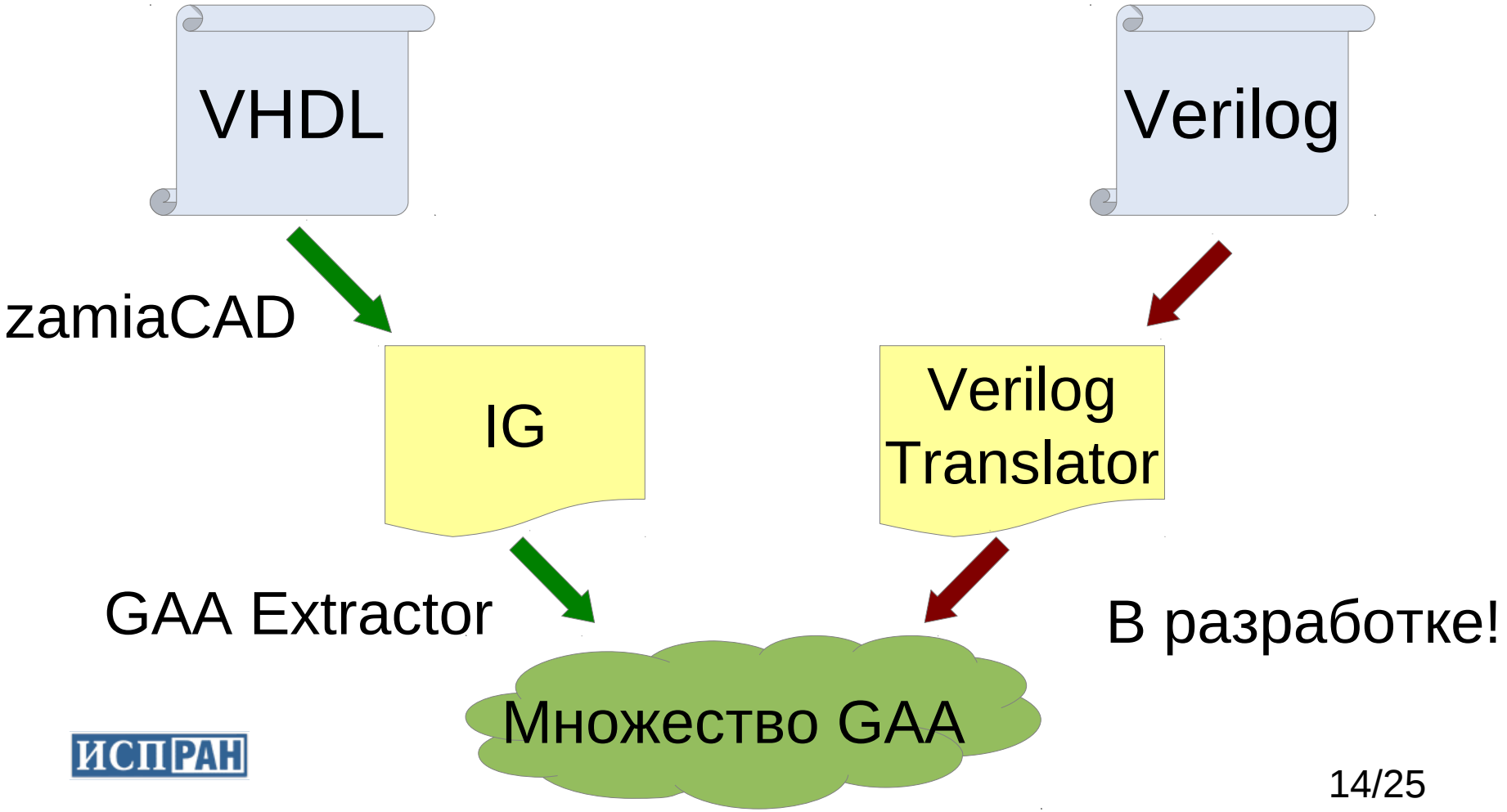
Идея метода генерации тестов

- Использовать единое внутреннее представление для HDL-описаний на Verilog и VHDL
- На основе анализа представления извлекать сведения:
  - о внутренней структуре (состояние)
  - об операциях
- Генерировать шаблоны тестов

Внутреннее представление

- Подход Clocked Guarded Actions (CGA), (University of Kaiserslautern, 2011)
  - G (C, ...)⟶A, где G – охранный предикат (Guard), A – выполняемые действия (*Actions*), C – сигнал синхронизации (*Clock*)
  - HDL-модель - <S,V>, где S – множество CGA, V – множество переменных
- Модификация – Guarded Atomic Actions (GAA)
  - G(...)⟶A
  - A – атомарное действие

Извлечение GAA из кода HDL-описаний



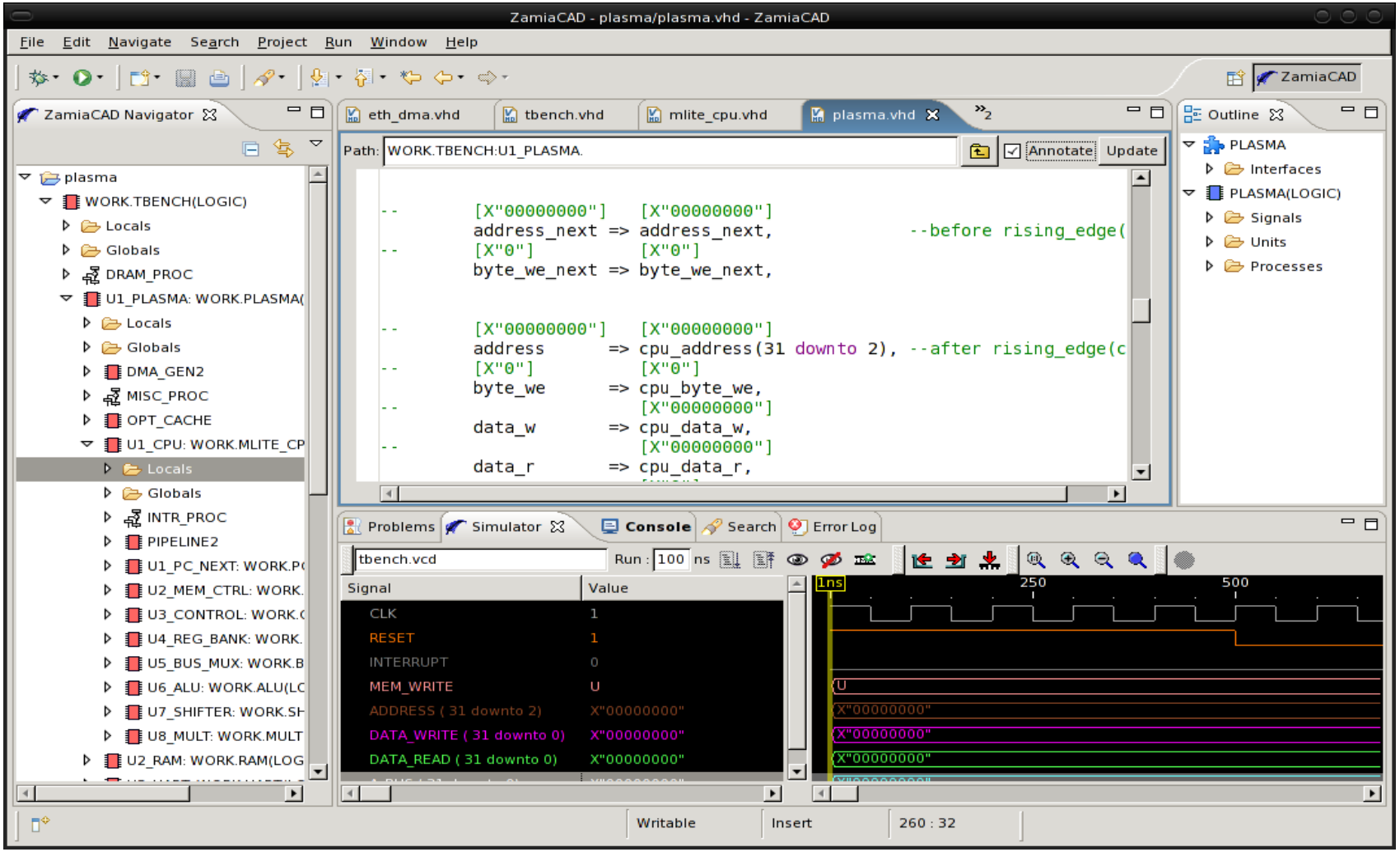
ИСП РАН

zamiaCAD

- Разработана в Tallinn University of Technology
- Расширяемая платформа для проектирования и анализа моделей аппаратуры
- Включает в себя:
  - **Внутреннее представление IG** (Instantiation Graph), не зависящее от HDL-языка
    - VHDL, Verilog (?)
  - Средства проектирования, анализа и симулирования

ИСП РАН

zamiaCAD

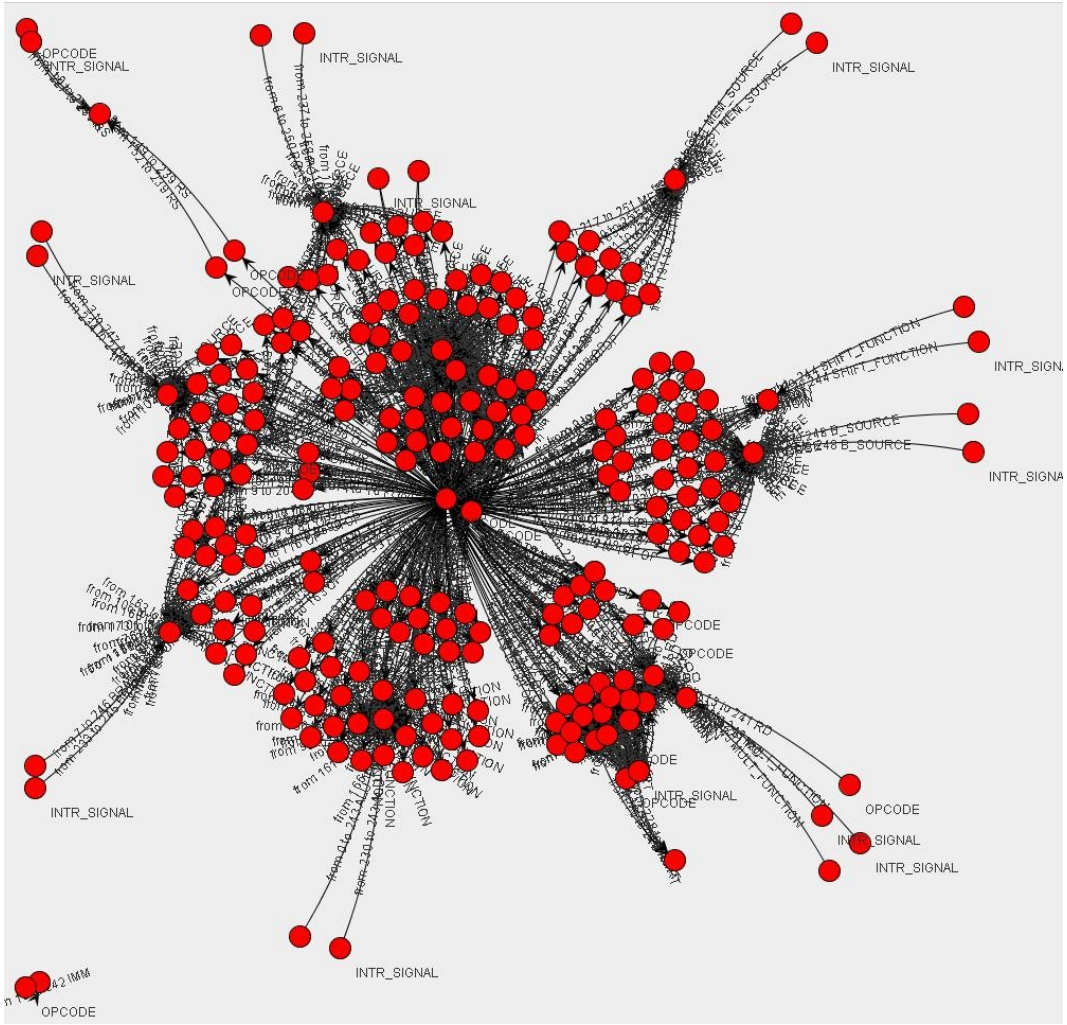


ИСП РАН

Зависимость по данным

- GA_1 (G1()⟶A1) и GA_2 (G2()⟶A2) являются **зависимыми по данным**, если найдется хотя бы одна переменная\сигнал, которой в одном из GAA присваивается значение, а в другом — её значение используется

Граф зависимостей по данным



- Вершины графа — GAA
- Ребра — зависимости по данным между ними

ИСП РАН

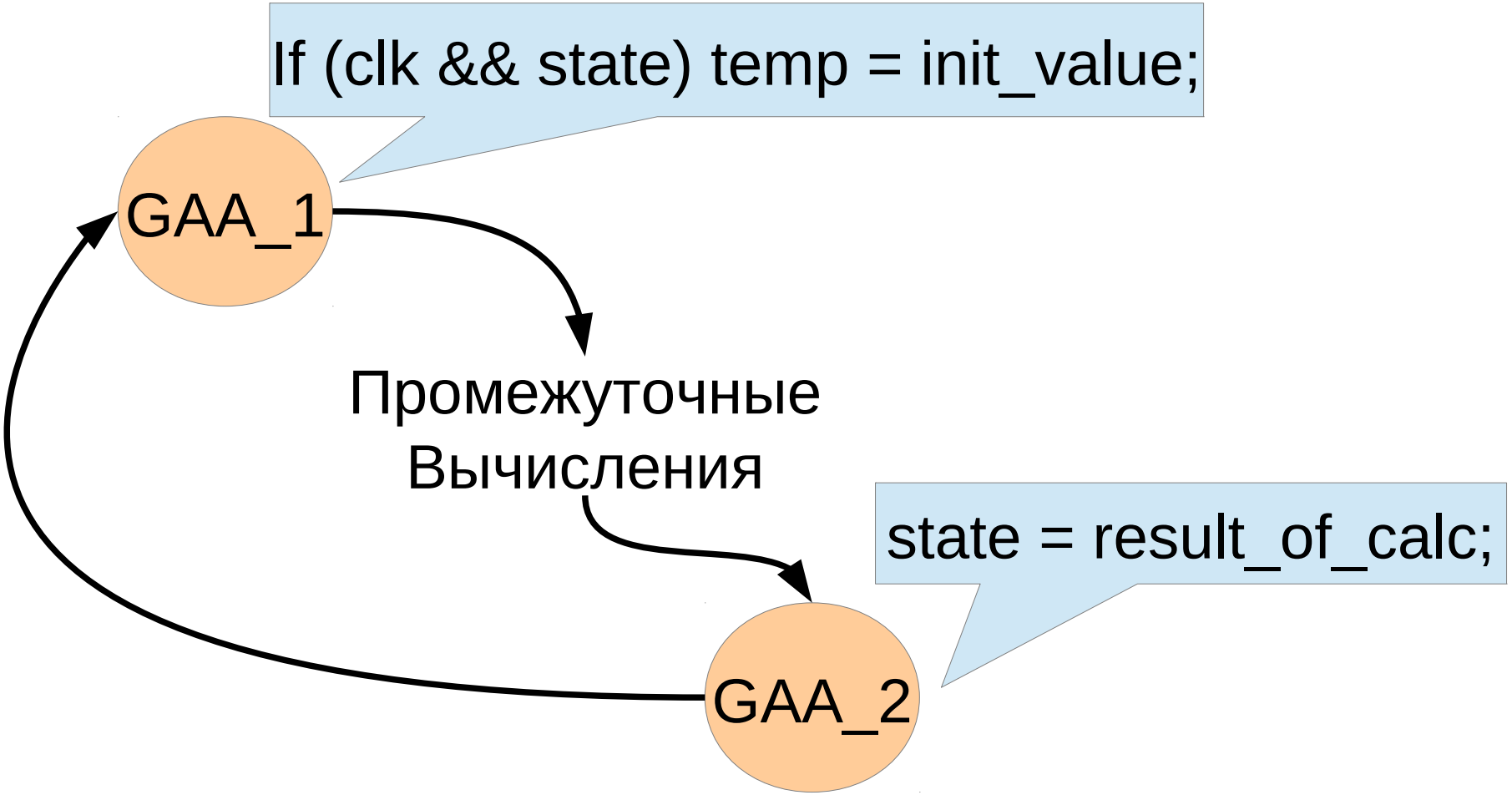
Метод извлечения состояния: определение 1

- **Часы**(clock) - это:
  - Входной сигнал модуля
  - Размерность = 1 бит
  - Значение не изменяется в присваиваниях

Метод извлечения состояния: определение 2

- **Переменная состояния** — это:
  - Не входной сигнал
  - В графе зависимостей по данным (DFG) найдется GAA_1, в G которого значение этой переменной используется, причем в множестве аргументов G есть хотя бы одни часы
  - В DFG найдется GAA_2, в котором этой переменной присваивается некоторое значение
  - В DFG есть путь из GAA_1 в GAA_2

Определение 2: иллюстрация



ИСПРАН

Метод извлечения состояния

- Преобразуем HDL-описание во внутреннее представление — множество GAA
- Строим граф зависимостей по данным
- На основе предложенных определений извлекаем набор переменных состояния

Текущие результаты

- Анализировался код 4 open-source VHDL проектов (14 000 строк кода, 42 модуля)
- На каждые 3 строки извлекался ~1 GAA
- Графы зависимостей — разреженные
- Для 10 модулей удалось извлечь состояние:
  - 30 % - *state
  - 70 % - «адекватные» переменные (size, address)

ИСП РАН

Ближайшие перспективы

- Извлечение сведений об операциях
- Вычисление областей допустимых значений
  - Граф потока управления\EFSM
  - Использование решателей (Z3 Solver)
- Генерация шаблонов тестов для C++TesK

Спасибо!

Вопросы?