Автоматизация процесса создания тестовых окружений, обеспечивающая сквозной маршрут разработки, верификации и исследования СФ-блоков и СнК

К.А. Жезлов, Я.С. Колбасов, А.О. Козлов, А.В. Николаев, Ф.М. Путря, С.Е. Фролова. ОАО НПЦ “ЭЛВИС”

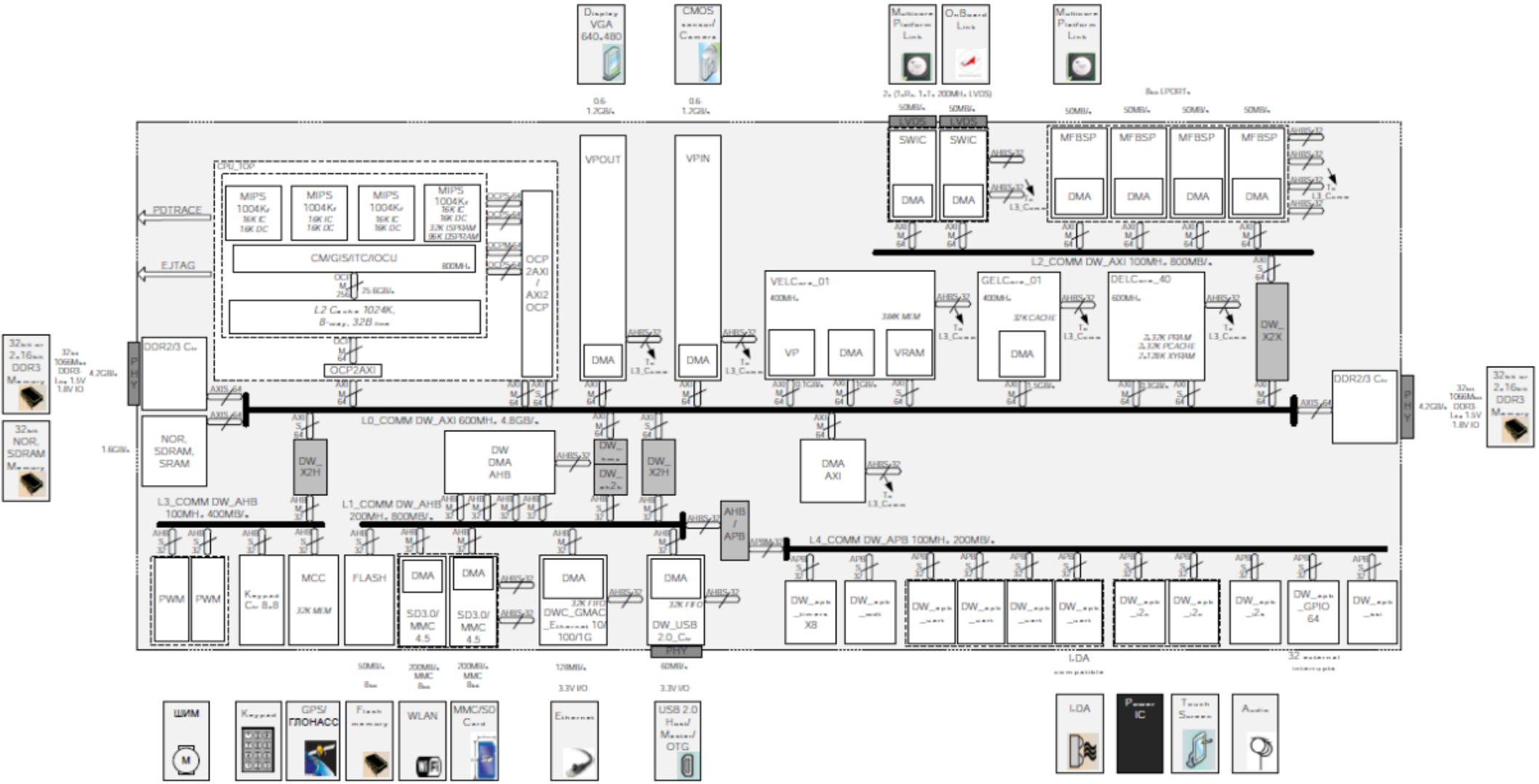
МЭС - 2016

Содержание

- Проблема переносимости верификационного кода между этапами верификации СнК
- Классификация методов создания тестовых воздействий
- Классификация СФ-блоков, для которых требуется создание тестовых окружений
- Генерация тестовых окружений на базе унифицированного окружения как способ снижения трудозатрат на создание и сопровождение большого числа окружений СФ-блоков в рамках проекта СнК
- Тестовое окружение как инструмент исследования
- Модульное средство сборки проектов project_compiler
- Заключение

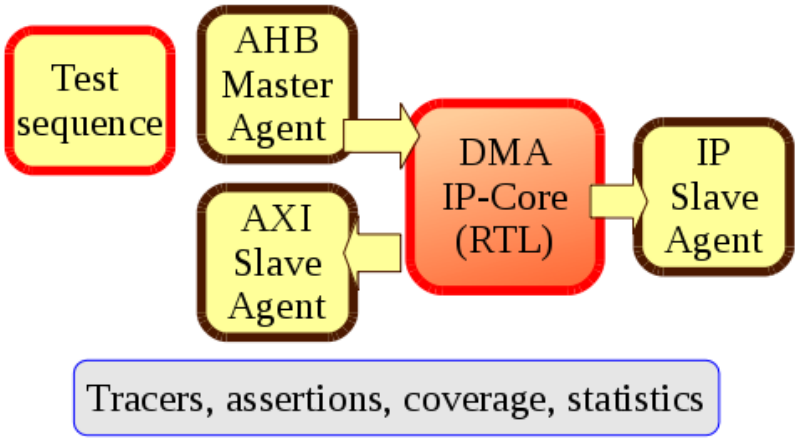
Содержание

Объект верификации - СнК

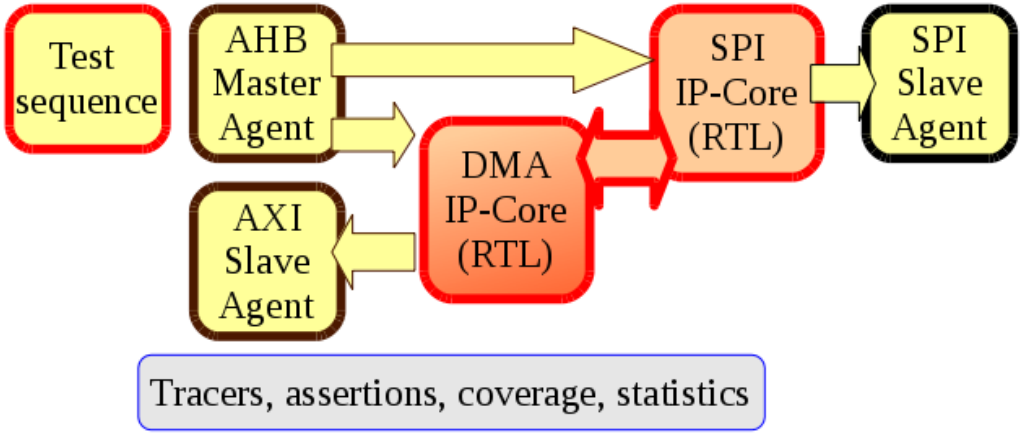


Стадии верификации СФ-блока

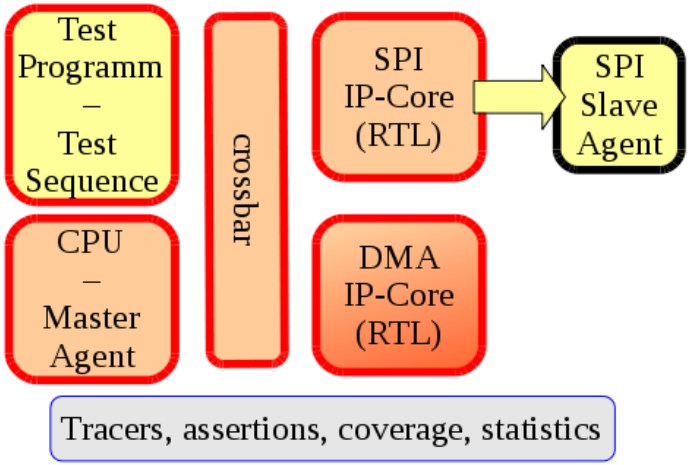
Автономное окружение СФ-блока



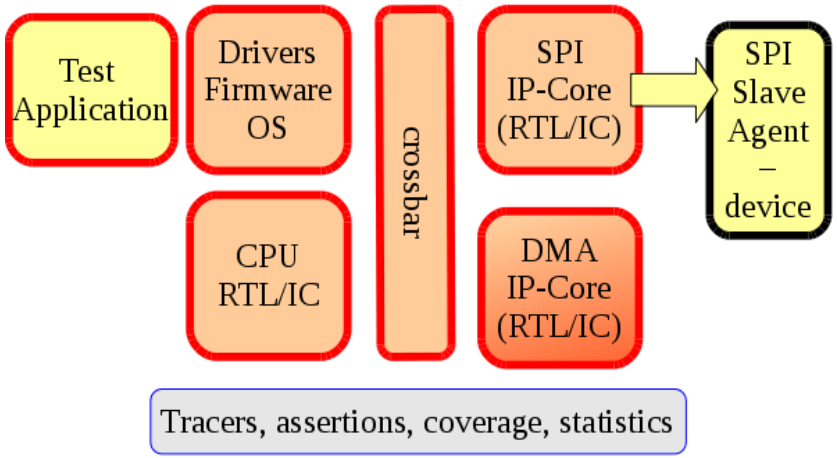
Автономное окружение группы IP-блоков



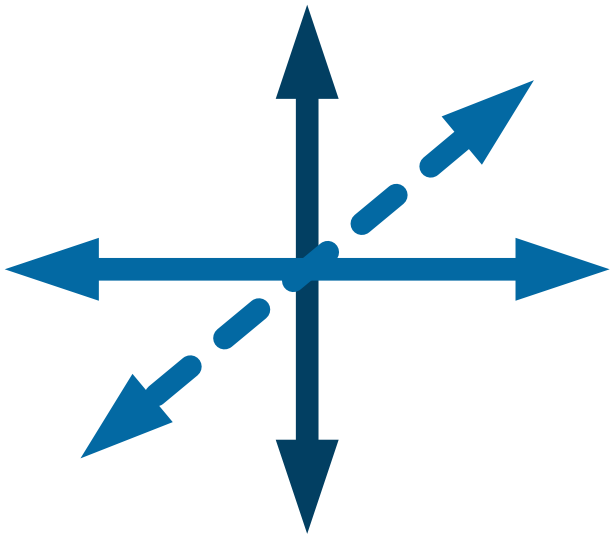
Тесты интеграции в СнК



Ко-верификация ПО и аппаратуры, Разработка драйверов и firmware



Многомерное пространство этапов верификации СнК



Измерения:

- Представление СФ-блока (TLM,RTL,STUB,netlist…)
- Уровень окружения (автономное, подсистема, система …)
- Способ оценки корректности модели (моделирование, эмуляция, прототипирование, формальная верификация
- САПР(Synopsys, Cadence, Mentor Graphics…)
- Проекты различных СнК или СФ-блоков

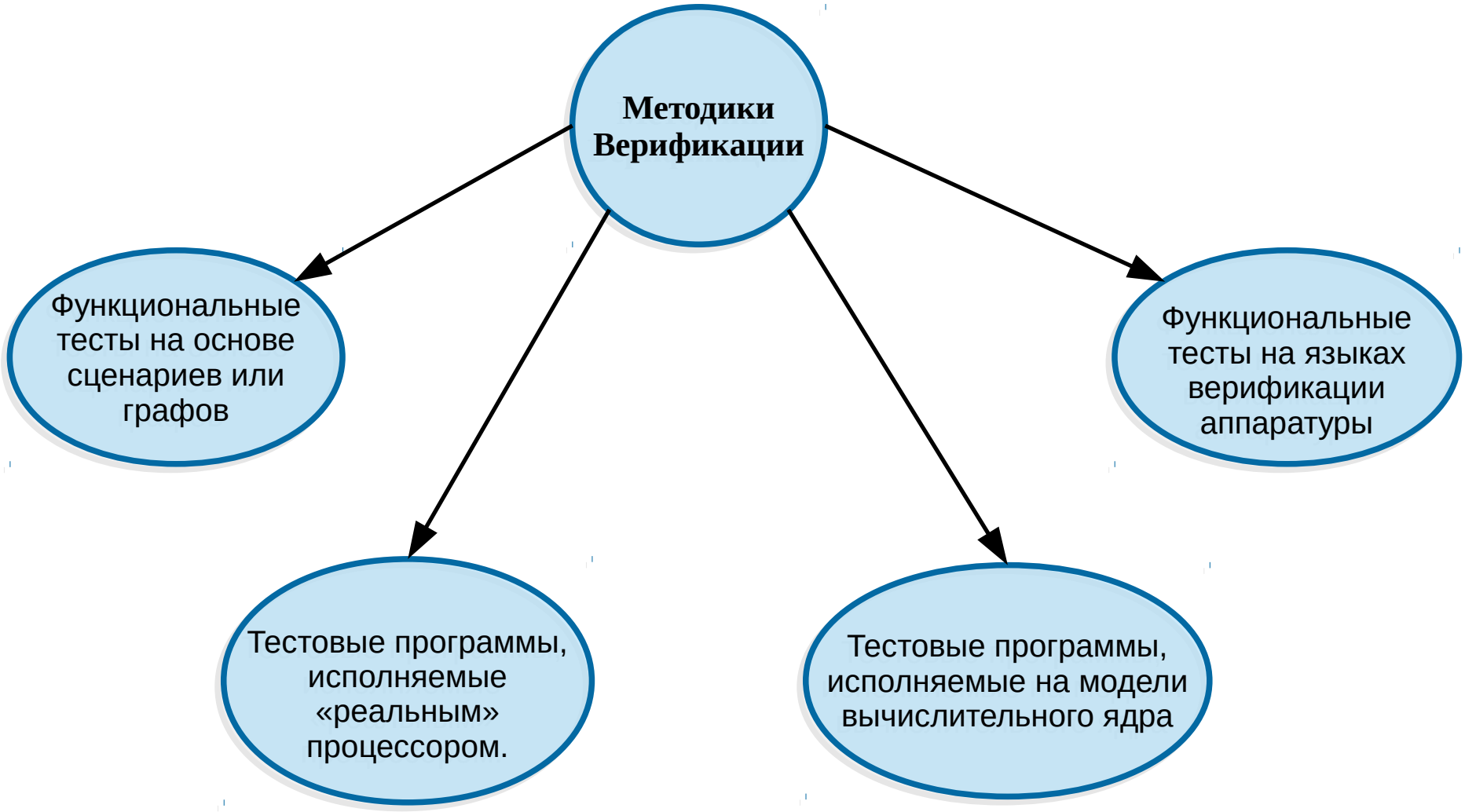
Пример точки в пространстве: верификация RTL-описания СФ-блока в автономном верификационном окружении путём симуляции средствами Cadence

Объекты, переносимость которых позволит решить задачу повышения эффективности труда инженеров

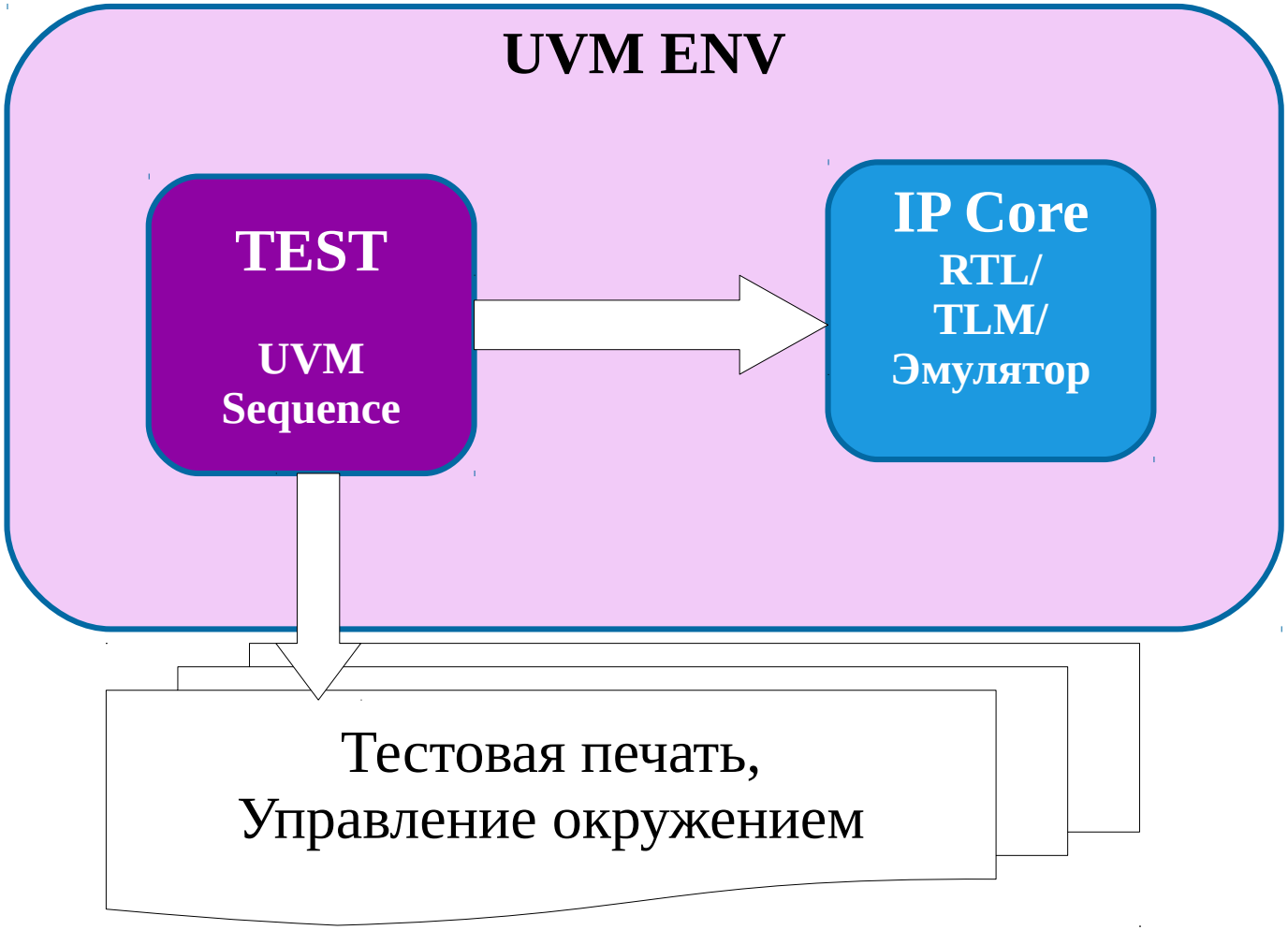
- Модели СФ-блоков (и средства их сборки)
- Программные драйверы СФ-блоков(и средства их сборки)
- Тесты и тестовые сценарии (и средства их сборки)
- **Верификационные компоненты и окружения (и средства их сборки)**

Содержание

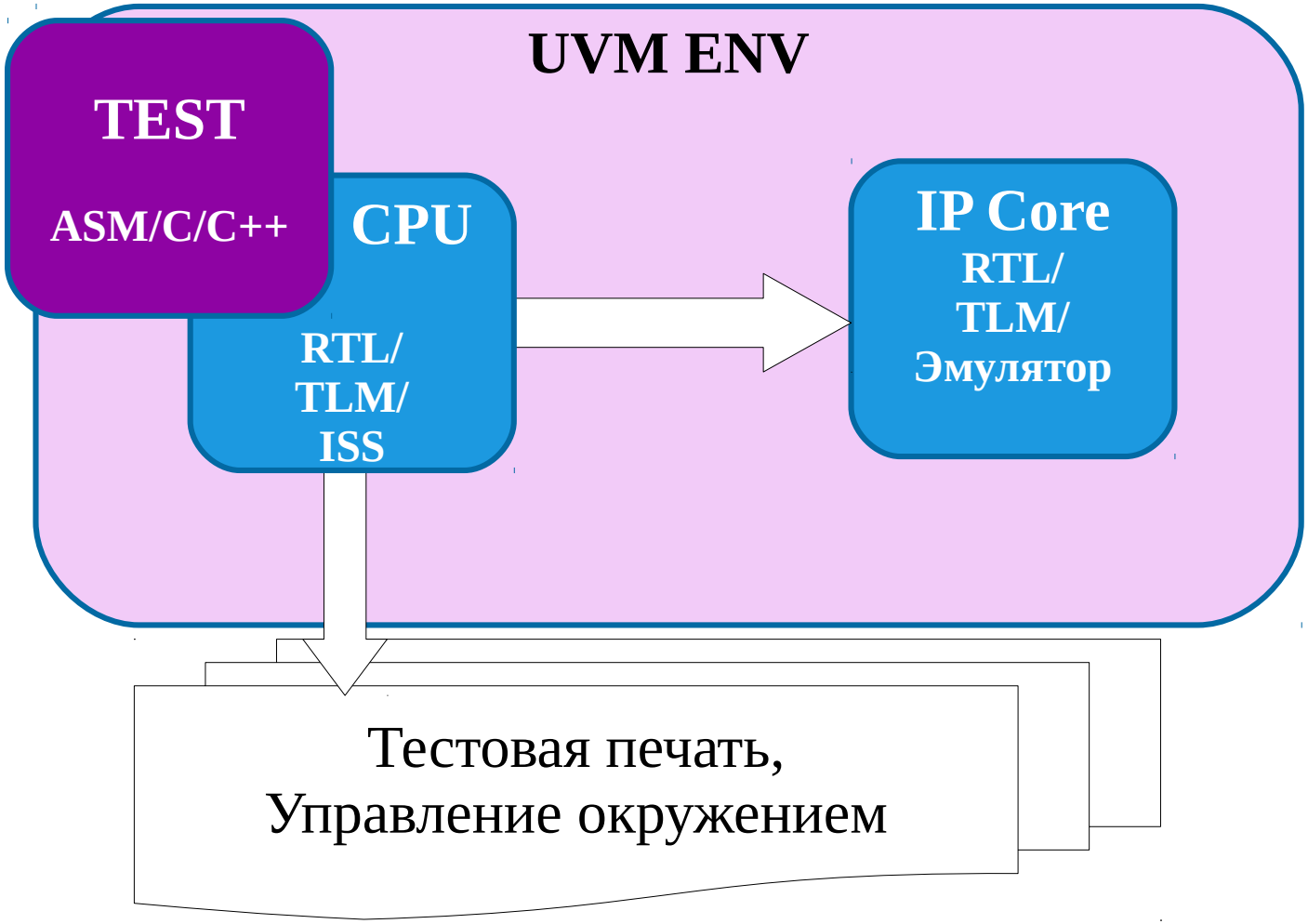
Классификация методов создания тестовых воздействий (динамическая верификация)



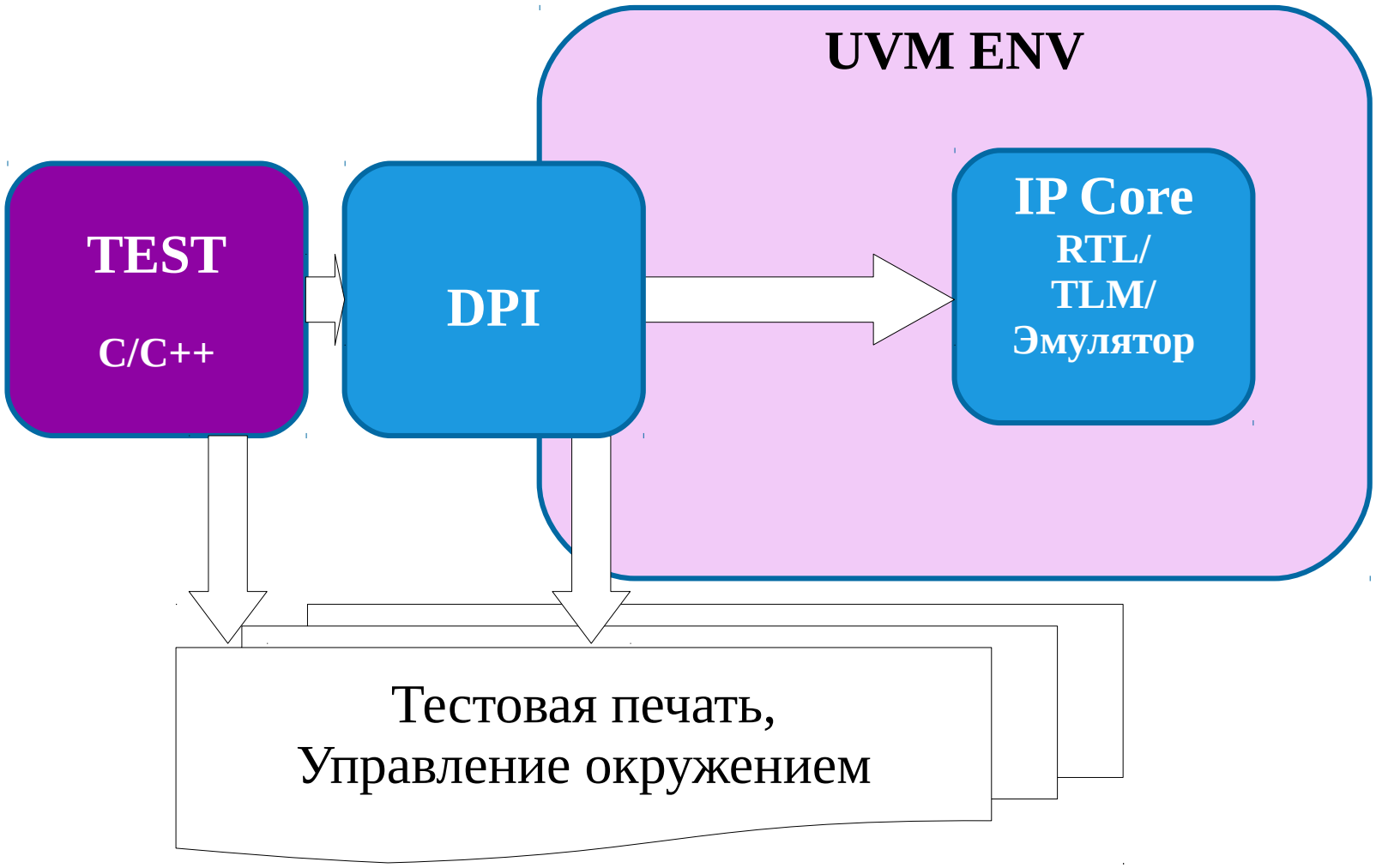
Функциональные тесты на языках верификации аппаратуры



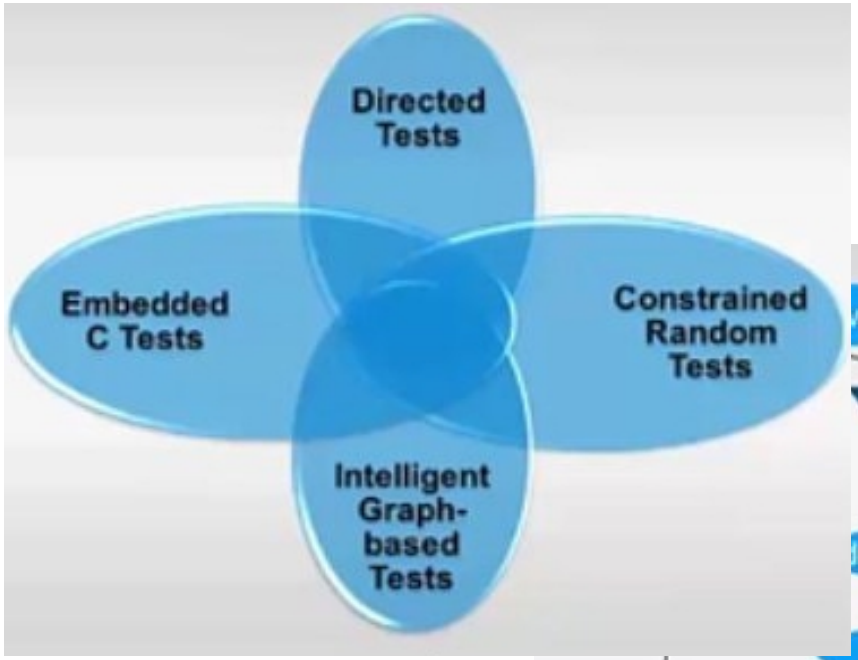
Функциональные тесты, исполняемые на модели вычислительного ядра



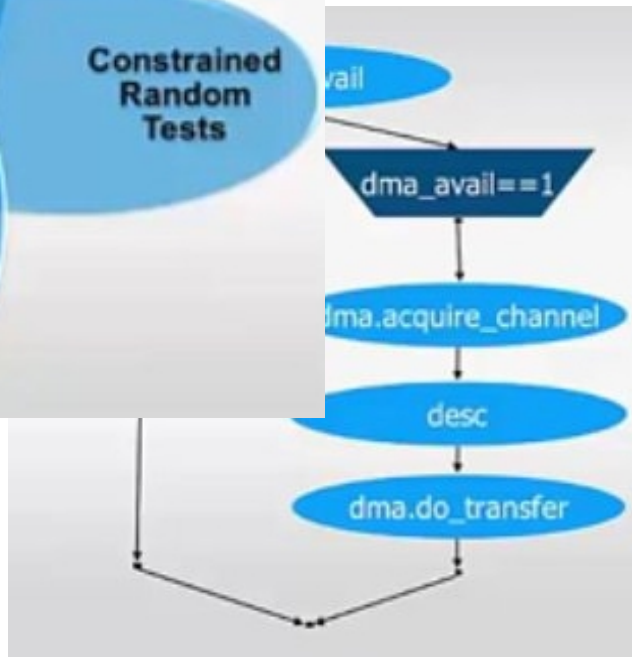
Функциональные тесты, исполняемые «реальным» процессором



Функциональные тесты на основе сценариев или графов



*** Verification Academy at DAC2015. Boosting Test-Creation Productivity with Portable Stimulus.**




```
graph mem_transfer(dma_if dma, mem_mgr_if mem) {
 rand dma_descriptor desc;
 int dma_avail, channel;
 mem_transfer(addr_t src, addr_t dst, size_t sz) := {
  dma.is_avail(dma_avail);
  select {
   if (dma_avail == 1) {
    dma.acquire_channel(channel);
    desc with {
     desc.addr_src == src;
     desc.addr_dst == dst;
     desc.tot_sz == sz;
    };
    dma.do_transfer(channel, desc);
   }
   mem.memcpy(dst, src, sz);
  } } }
```

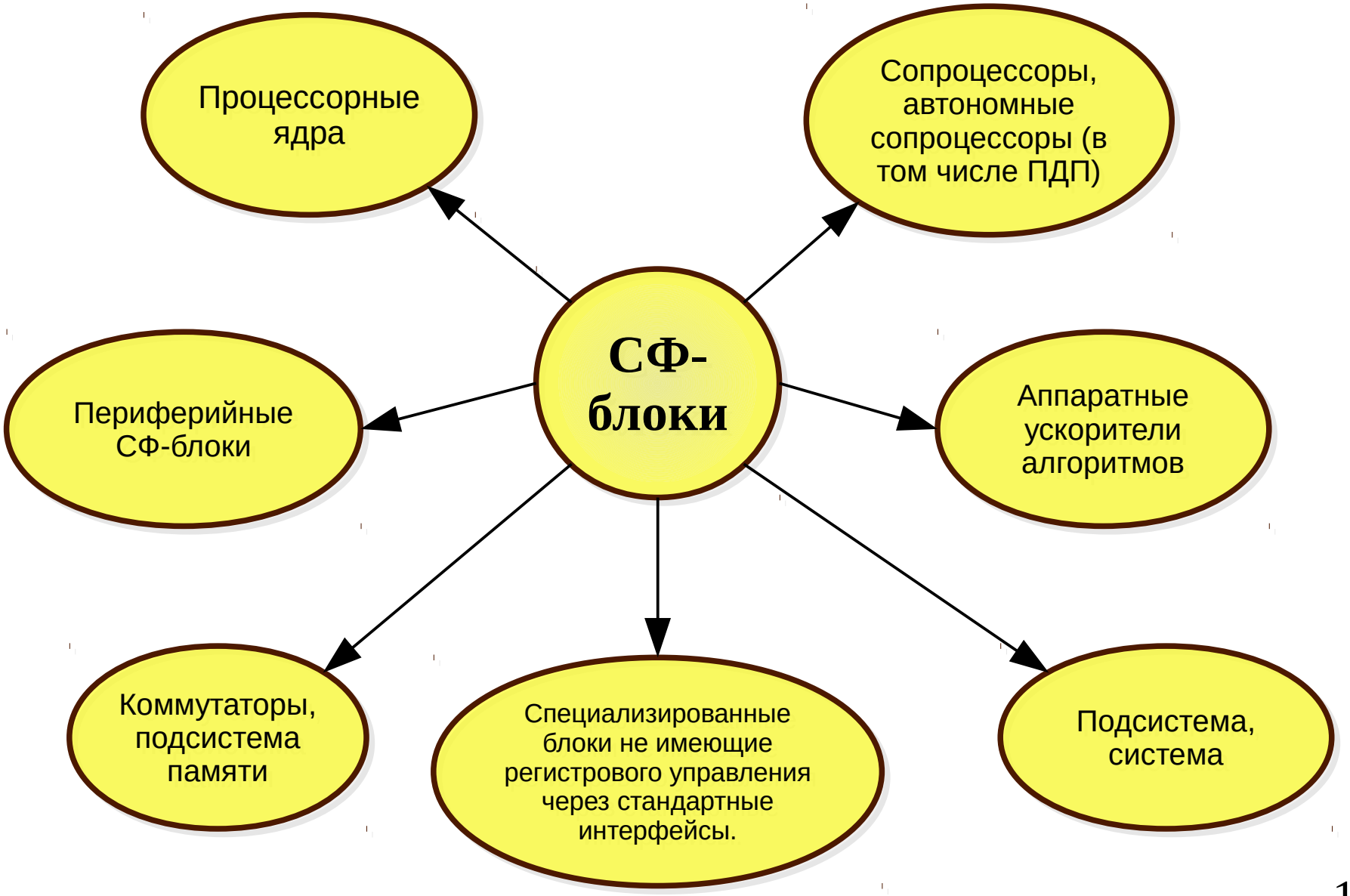
accellera SYSTEMS INITIATIVE

Accellera Portable Stimulus Working Group

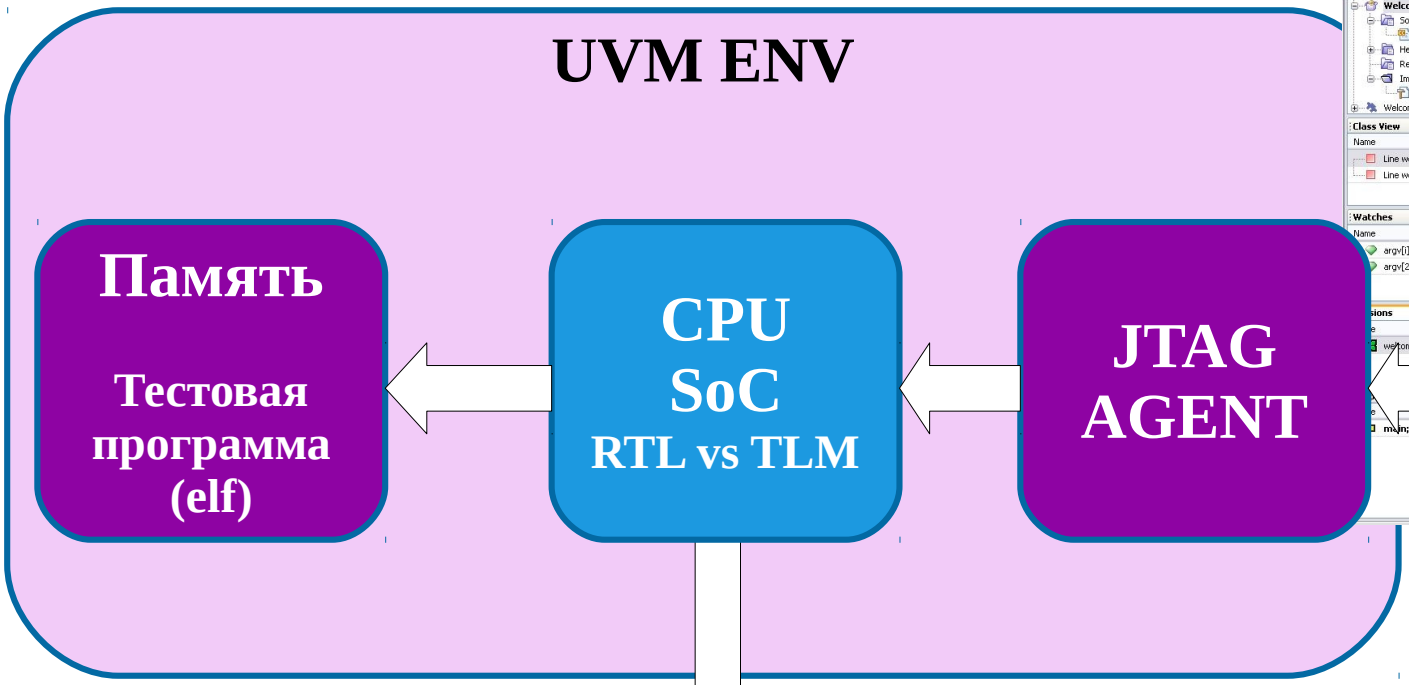
Содержание

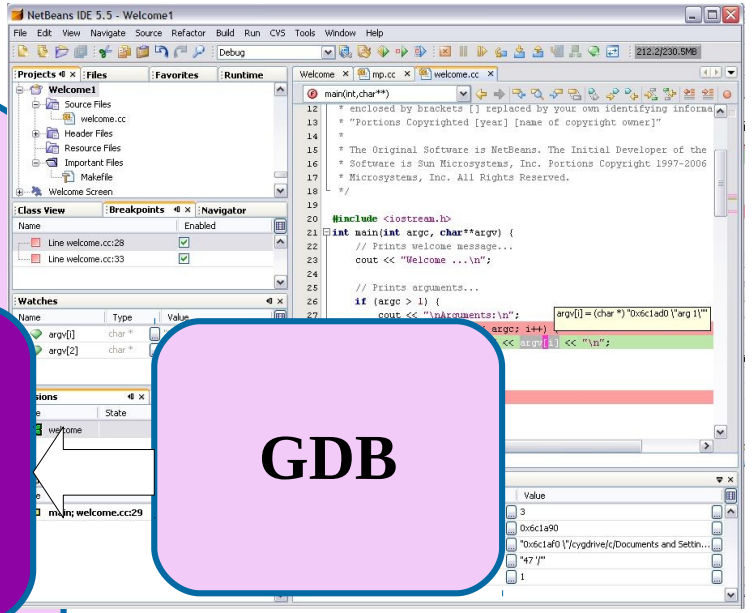
- Проблема переносимости верификационного кода между этапами верификации СнК
- Классификация методов создания тестовых воздействий
- **Классификация СФ-блоков, для которых требуется создание тестовых окружений**
- Генерация тестовых окружений на базе унифицированного окружения, как способ снижения трудозатрат на создание и сопровождение большого числа окружений СФ-блоков в рамках проекта СнК
- Тестовое окружение как инструмент исследования
- Модульное средство сборки проектов project_compiler
- Заключение

Классификация СФ-блоков



Тестовые окружения процессоров и СнК

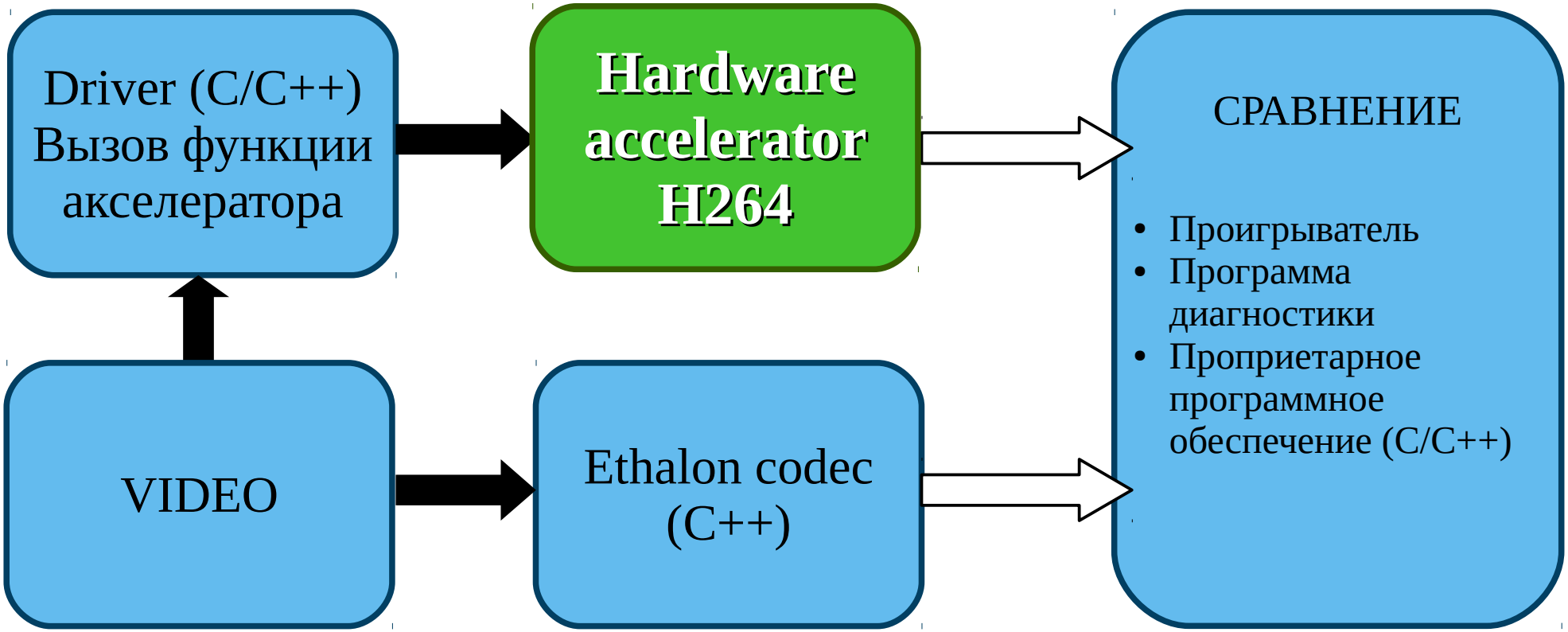




•Тесты запускаются на модели ядра.

•Минимальная требуемая оснастка: ведомые интерфейсы, модель памяти, эмуляция системы отладки, интерфейс взаимодействия программы ядра и окружения

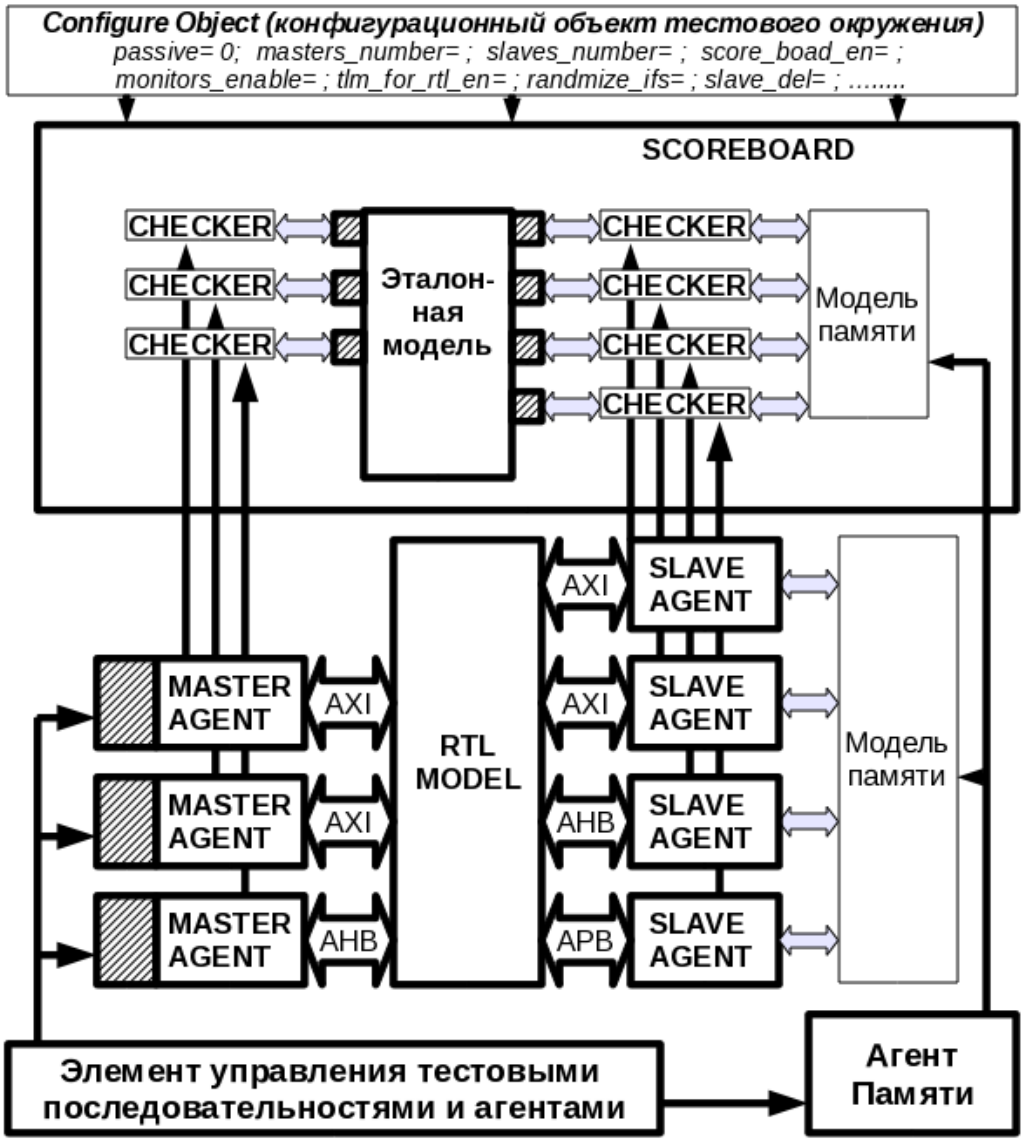
Аппаратные ускорители алгоритмов



•Тесты и эталоны — уже готовые (полностью или частично) программы на C/C++

•В зависимости от модели управления устройством сложность программного драйвера может варьироваться

Коммутаторы, подсистема памяти



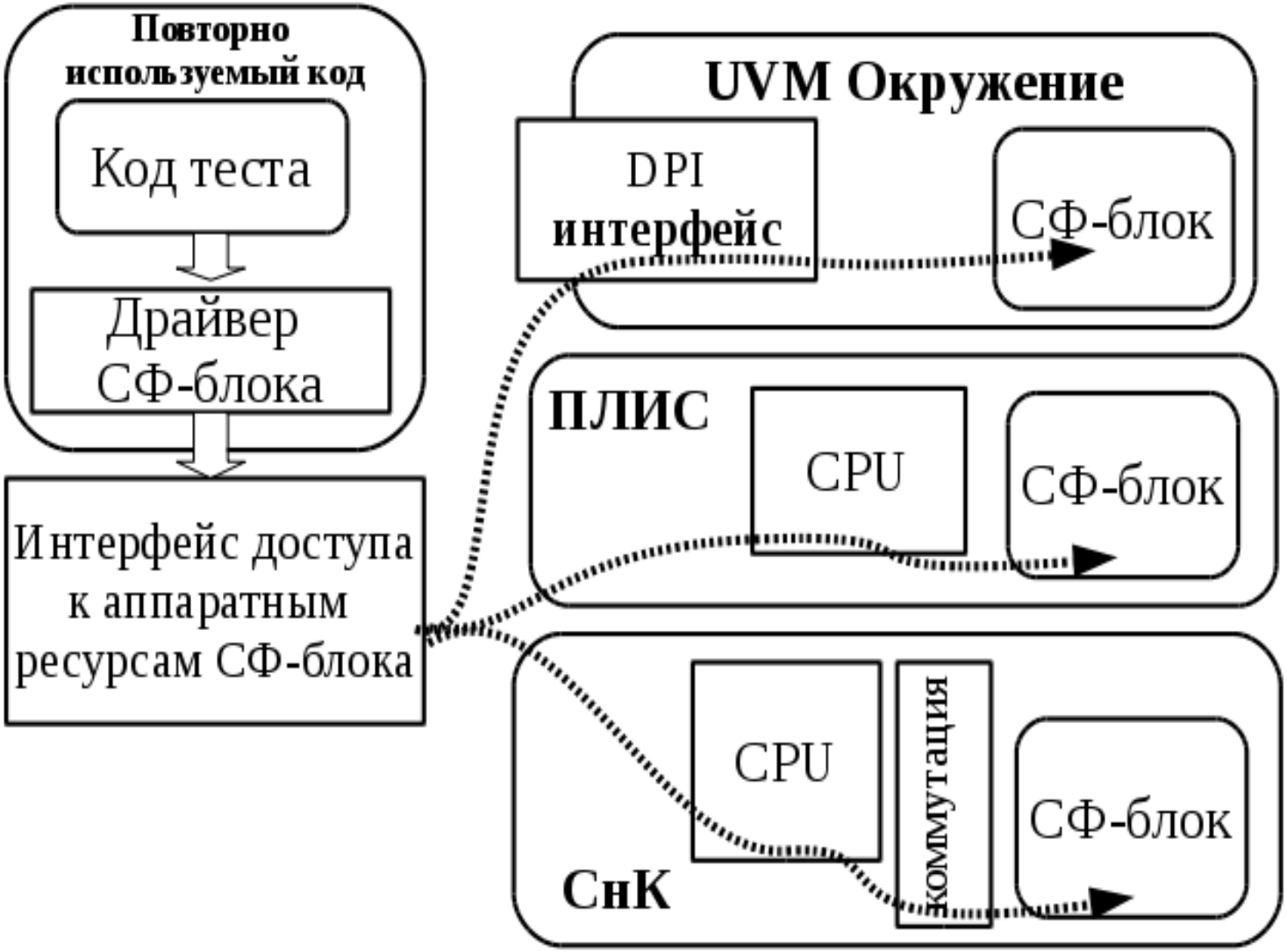
В одном устройстве:

- Разные типы интерфейсов
- Интерфейсы одного типа, с разными параметрами (ширины шин данных, идентификаторов, адреса и т.п.)
- Для переносимости тестов применено многослойное окружение и генерация тестов на уровне абстракции не привязанной к конкретным интерфейсам

Содержание

- Проблема переносимости верификационного кода между этапами верификации СнК
- Классификация методов создания тестовых воздействий
- Классификация СФ-блоков, для которых требуется создание тестовых окружений
- **Генерация тестовых окружений на базе унифицированного окружения, как способ снижения трудозатрат на создание и сопровождение большого числа окружений СФ-блоков в рамках проекта СнК**
- Тестовое окружение как инструмент исследования
- Модульное средство сборки проектов project_compiler
- Заключение

Запуск тестов на окружении блока, системы и прототипе

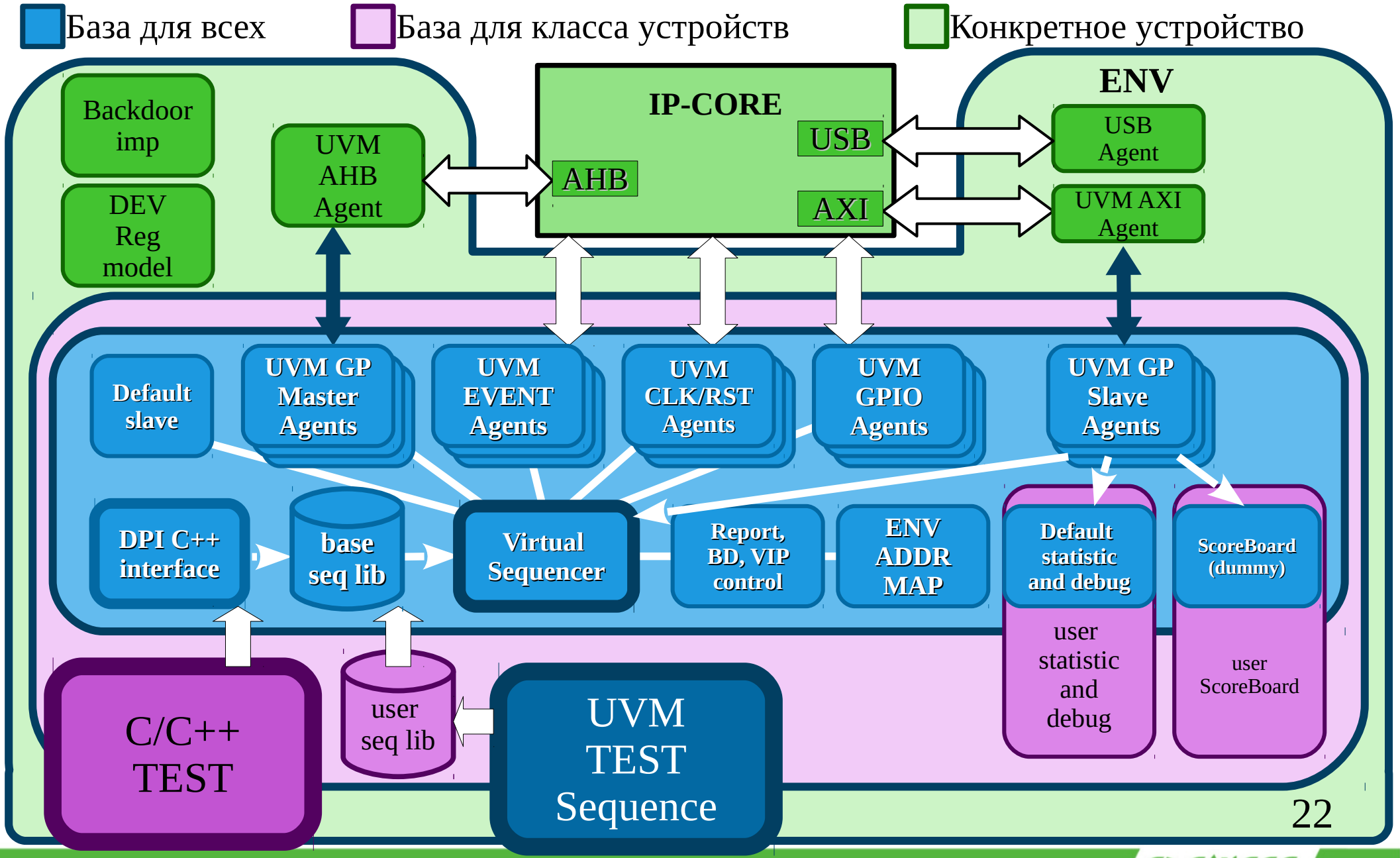


•Сложные драйверы и тесты устройств переносятся между этапами верификации без изменений

Ключевые особенности генерируемых тестовых окружений

- **Обеспечение возможности запуска сценариев на SV, C++ (рабочая станция и/или модель процессора) или считываемых из текстового файла со сценарием.**
- **Единая для автономных и системных окружений инфраструктура тестовой печати, управления внешними агентами (VIP) и синхронизации между процессами, доступная всем типам тестовых сценариев.**
- **Единое адресное пространство, доступное со стороны тестовых сценариев, верифицируемого устройства и агентов быстрого доступа к памяти**
- **Интегрированные агент JTAG интерфейса и средства подключения стандартного отладчика (типа gdb).**
- **Параметризация верификационных компонент**
- **Возможность запуска высокоуровневой (C++/TLM) модели устройства.**
- **Поддержка пассивного режима работы окружения.**
- **Все агенты интерфейсов содержат мосты для преобразования транзакций к унифицированному типу**

Унифицированное тестовое окружение

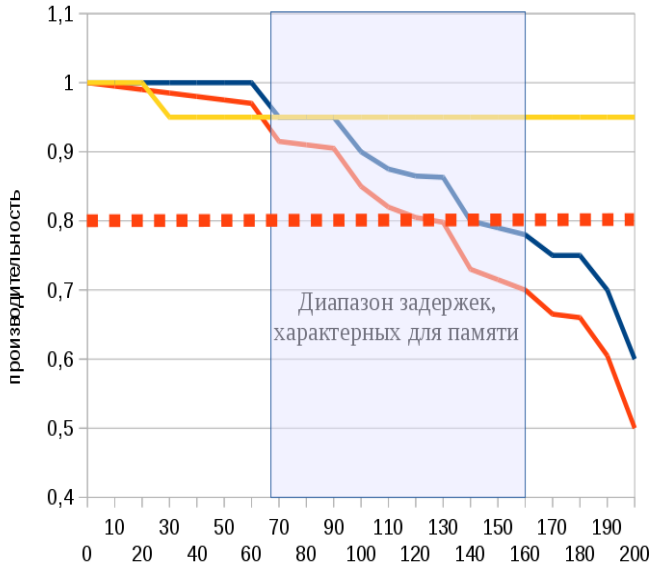


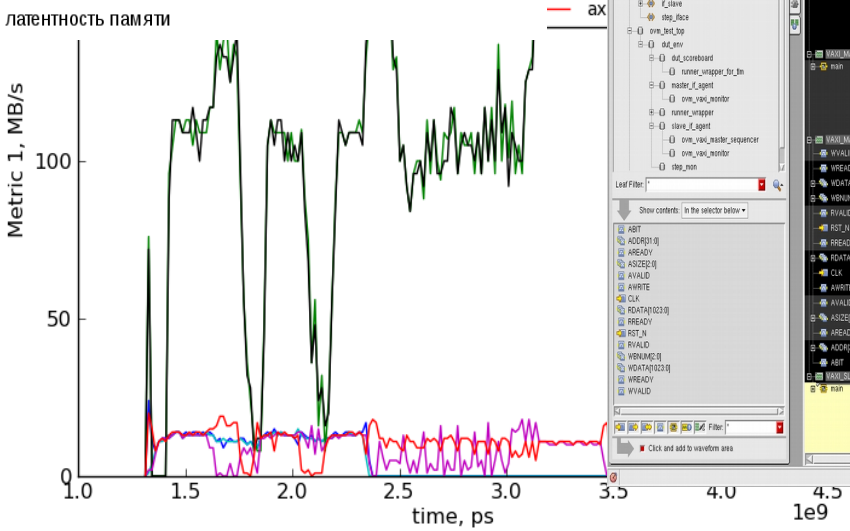
Содержание

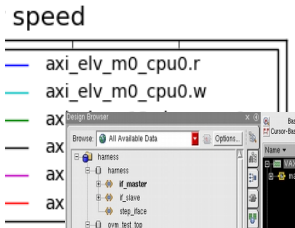
- Проблема переносимости верификационного кода между этапами верификации СнК
- Классификация методов создания тестовых воздействий
- Классификация СФ-блоков, для которых требуется создание тестовых окружений
- Генерация тестовых окружений на базе унифицированного окружения, как способ снижения трудозатрат на создание и сопровождение большого числа окружений СФ-блоков в рамках проекта СнК
- **Тестовое окружение как инструмент исследования**
- Модульное средство сборки проектов project_compiler
- Заключение

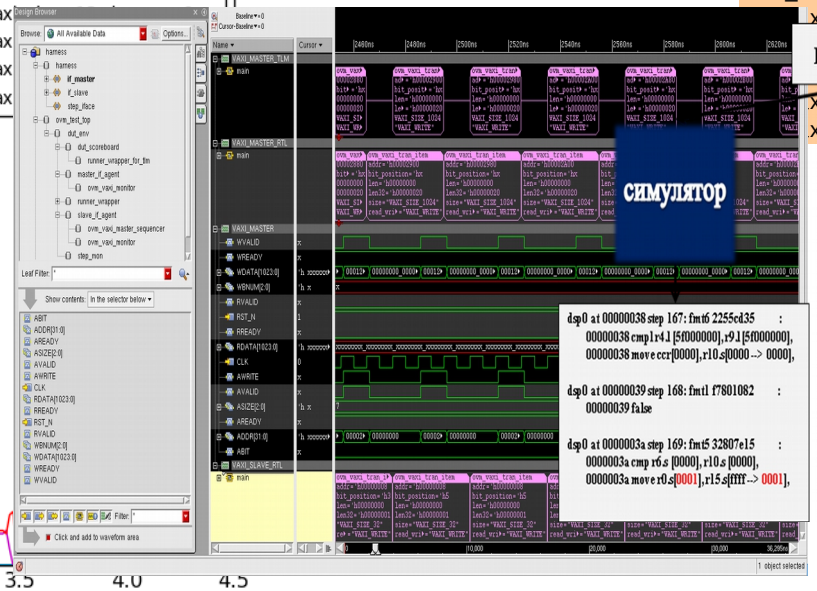
Унифицированные инструменты анализа статистики, метрик и отладки

унифицированное окружение блока + средства трассировки









весь тест	такты
data_axi_elv_m14_dma_w_m	2
data_axi_elv_m13_dma_r_m0	7
data_axi_elv_m5_nand0.w	11
data_axi_elv_m11_nand6.r	11
data_axi_elv_m9_nand4.r	11
data_axi_elv_m0_cpu0.r	13
data_axi_elv_m8_nand3.w	14
data_axi_elv_m6_nand1.w	14
data_axi_elv_m12_nand7.r	14
data_axi_elv_m10_nand5.w	14
data_axi_elv_m7_nand2.w	15
data_axi_elv_m9_nand4.w	17
data_axi_elv_m12_nand7.w	17
data_axi_elv_m11_nand6.w	17
data_axi_elv_m5_nand0.r	19
data_axi_elv_m7_nand2.r	20
data_axi_elv_m10_nand5.r	20
data_axi_elv_m6_nand1.r	22
data_axi_elv_m8_nand3.r	23
data_axi_elv_s0_ddr0.r	37
data_axi_elv_s1_ddr1.r	37
xi_elv_m1_cpu1.r	41
1.w	43
0.w	43
xi_elv_m1_cpu1.w	44
xi_elv_s0_ddr0.w	44

программа

RTL

автоматическое сравнение

Содержание

- Проблема переносимости верификационного кода между этапами верификации СнК
- Классификация методов создания тестовых воздействий
- Классификация СФ-блоков, для которых требуется создание тестовых окружений
- Генерация тестовых окружений на базе унифицированного окружения, как способ снижения трудозатрат на создание и сопровождение большого числа окружений СФ-блоков в рамках проекта СнК
- Тестовое окружение как инструмент исследования
- **Модульное средство сборки проектов project_compiler**
- Заключение

Проверка кода на соответствие требованиям со стороны средств автоматизации перед загрузкой в библиотеку.

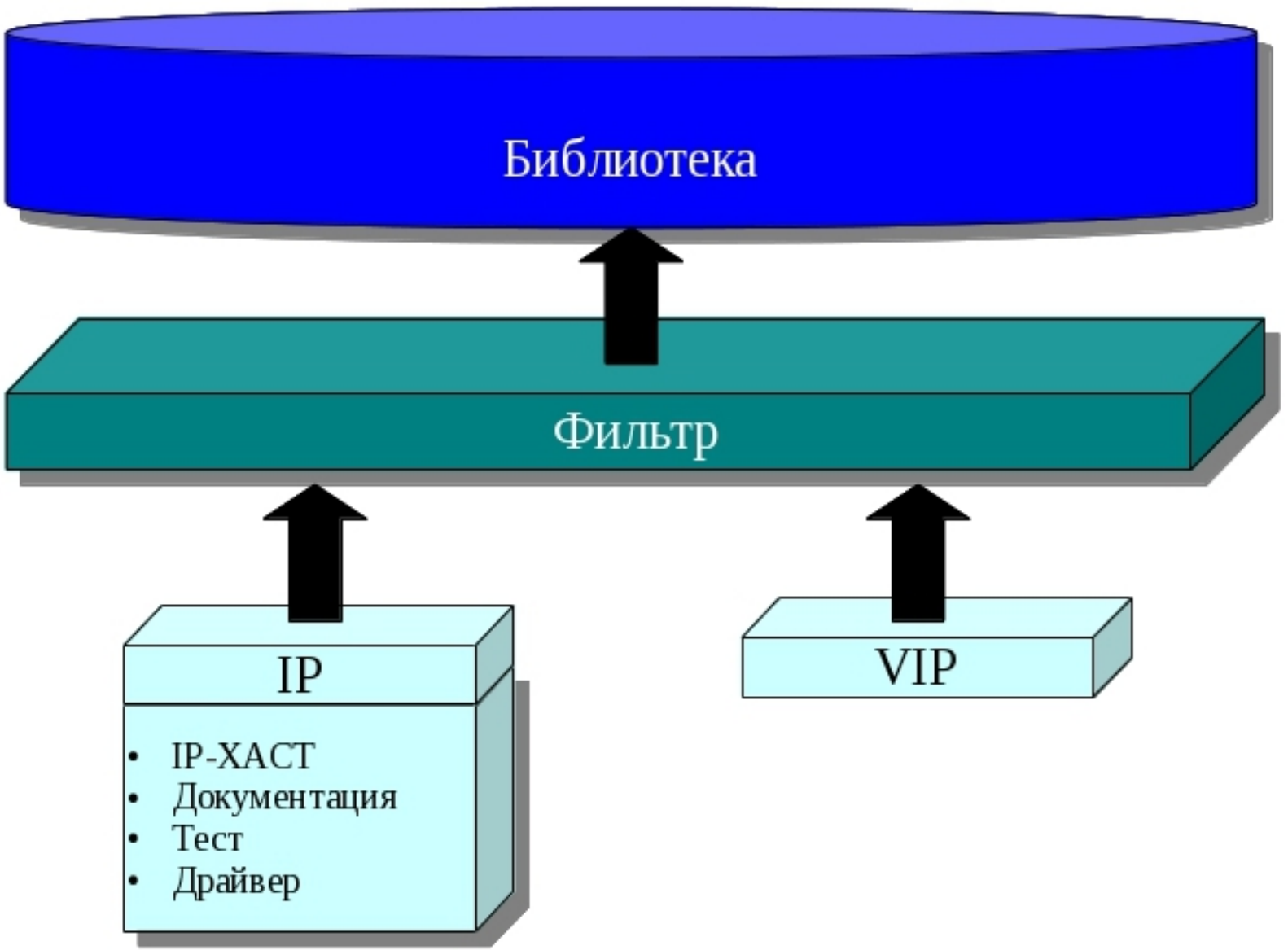
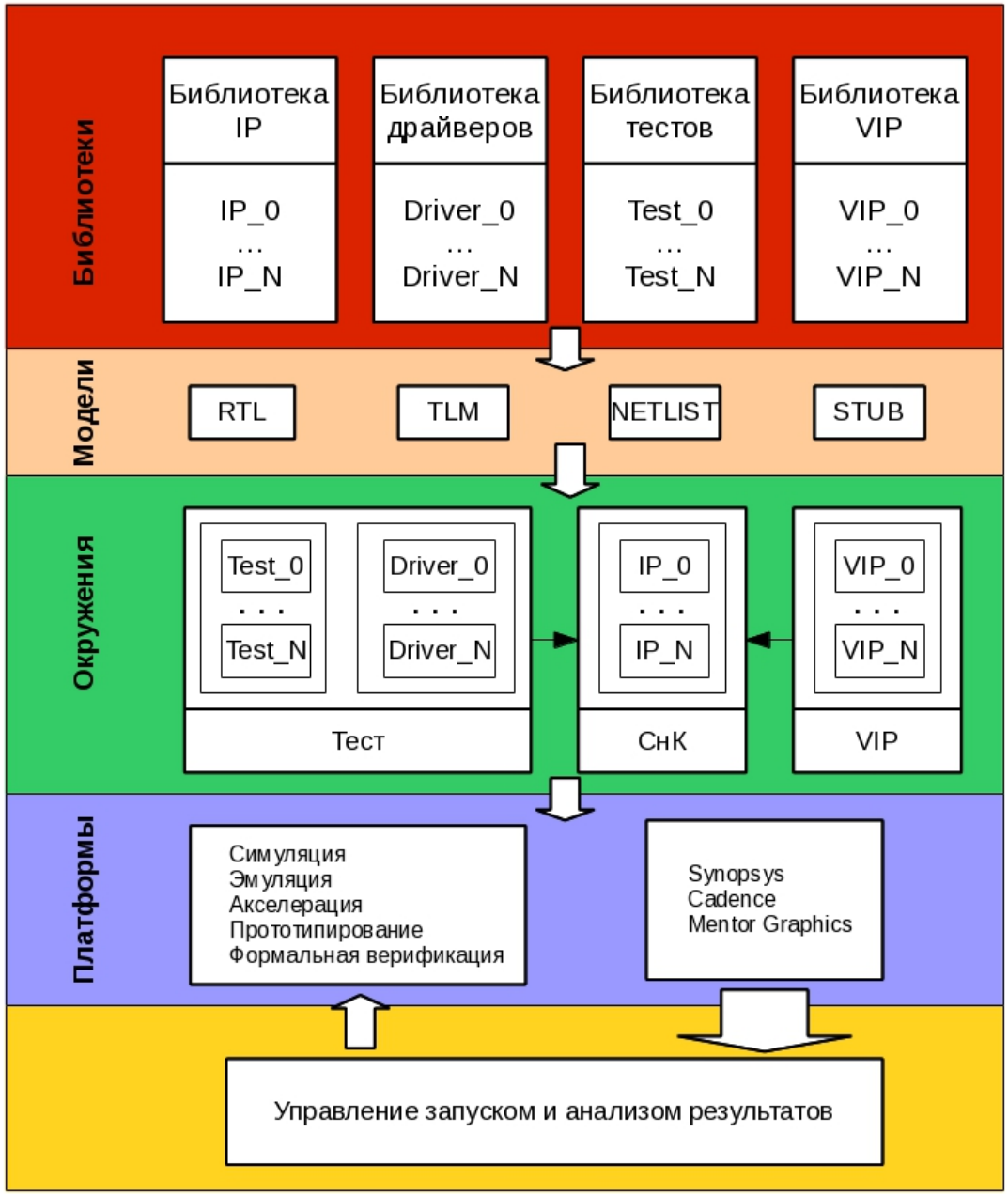
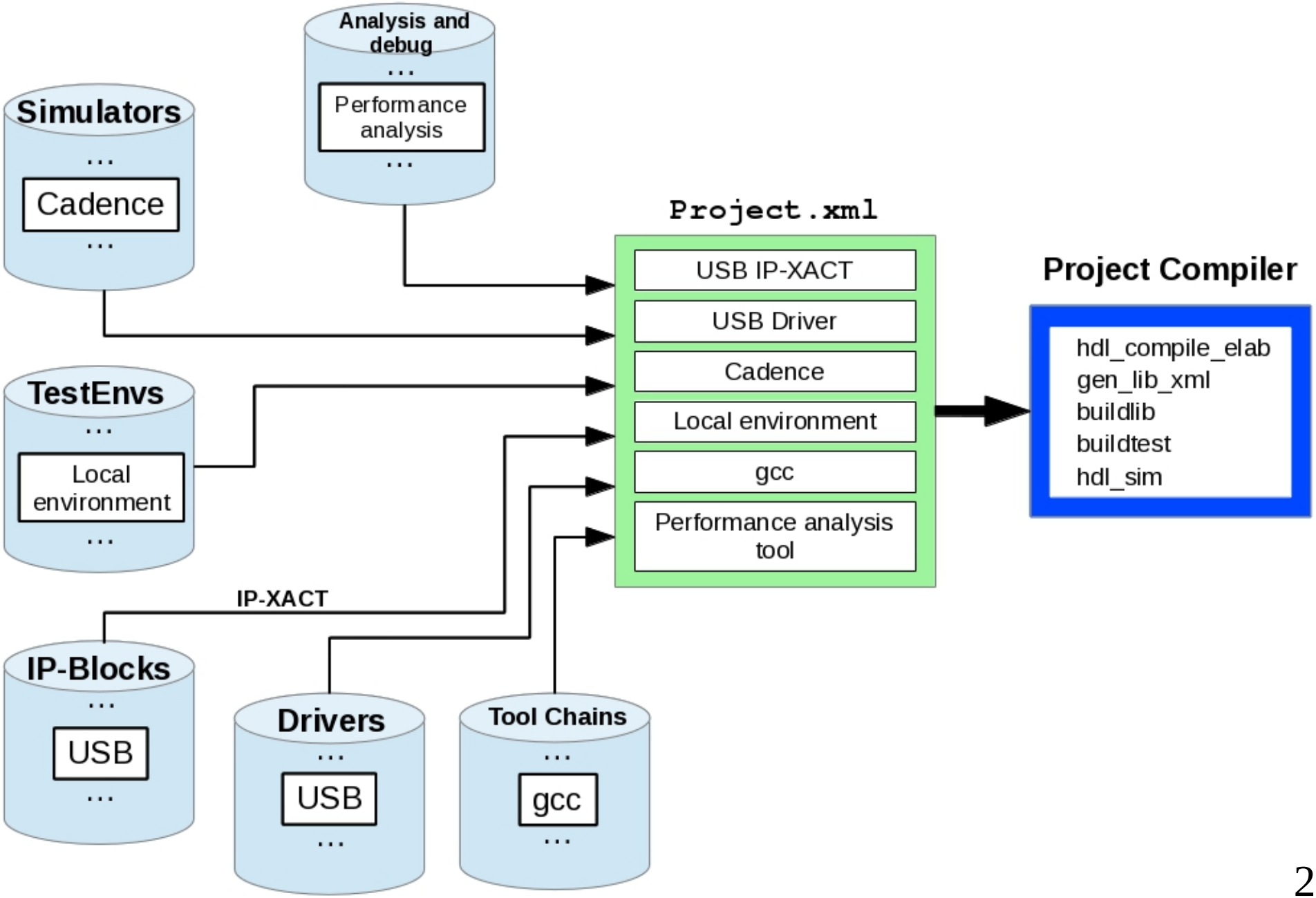


Схема процесса сборки инфраструктуры для выполнения этапа верификации блока/СнК



Конструктор верификационных средств project_compiler



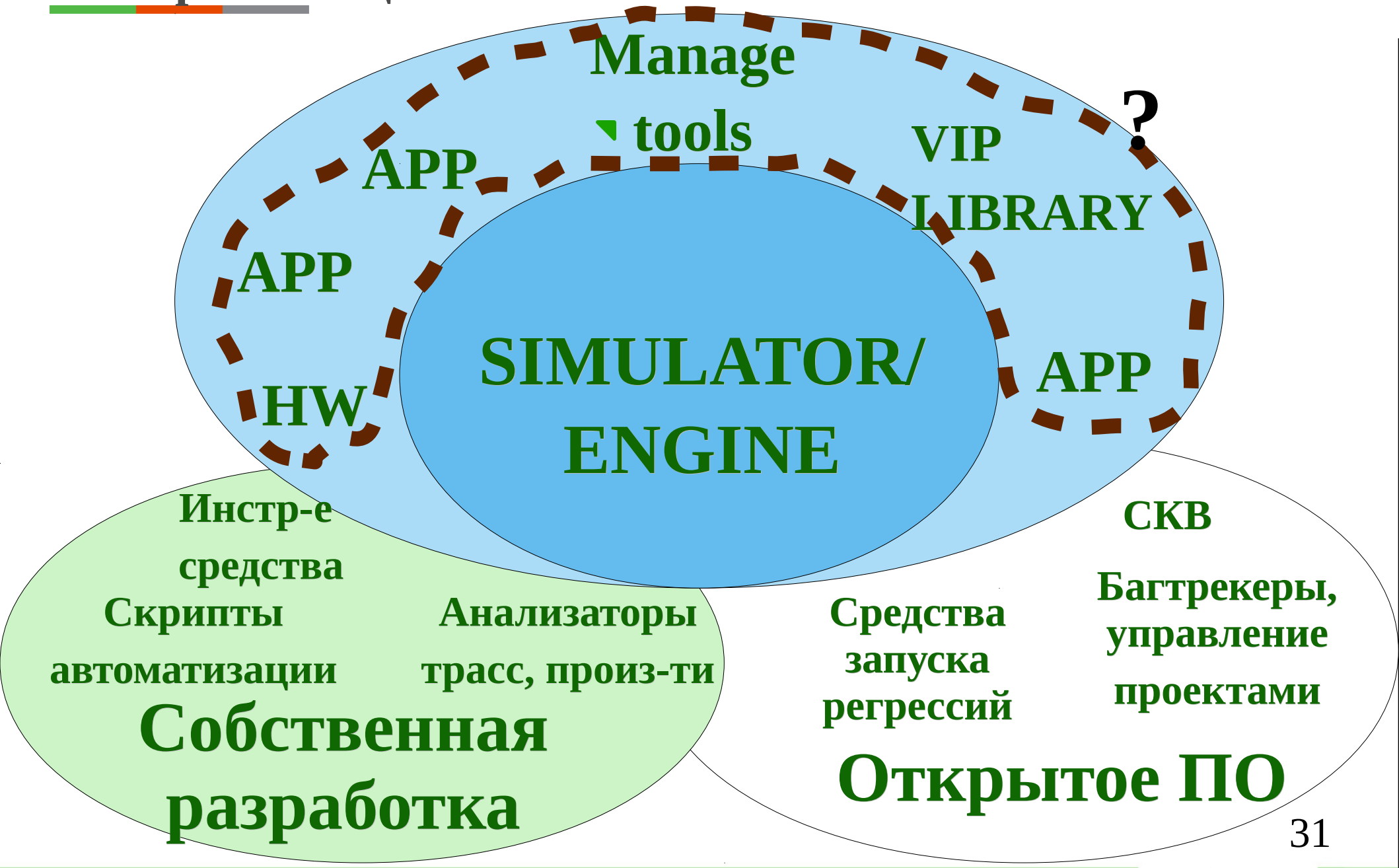
Содержание

- Проблема переносимости верификационного кода между этапами верификации СнК
- Классификация методов создания тестовых воздействий
- Классификация СФ-блоков, для которых требуется создание тестовых окружений
- Генерация тестовых окружений на базе унифицированного окружения, как способ снижения трудозатрат на создание и сопровождение большого числа окружений СФ-блоков в рамках проекта СнК
- Тестовое окружение как инструмент исследования
- Модульное средство сборки проектов project_compiler
- **Заключение**

Заключение

- За счёт автоматизации процесса создания верификационной инфраструктуры ускорен и упрощён процесс создания окружений для СФ-блоков, минимизирован процесс адаптации разработчика тестов к окружению нового СФ-блока и упрощен перенос тестов между окружениями однотипных блоков.
- Благодаря возможности исполнения тестовых программ на автономных окружениях СФ-блоков, обеспечена переносимость тестов между автономными и системными окружениями.
- За счёт унификации выходной информации, генерируемой тестовыми окружениями,создать единую инфраструктуру для исследования результатов тестов и характеристик СФ-блоков.
- Обеспечена централизация знаний таким образом, что все новые исправления, улучшения и утилиты автоматически становятся доступны всем, использующим тестовые окружения, созданные на базе шаблонного, что упрощает поддержание верификационной инфраструктуры.
- Обеспечено быстрое создание тестовой обвязки для окружений системного уровня
- Стандартизировано взаимодействие групп разработчиков, верификаторов, топологов и др. в рамках предприятия, а также формализованы и автоматизированы многие этапы проектирования и верификации систем на кристалле, что положительно сказалось на сроках выполнения проектов.

Инфраструктура для верификации проектов
Импортозамещение САПР?



Спасибо за внимание!

Контакты
www.multicore.ru
support@elvees.com
(495) 913-32-51