



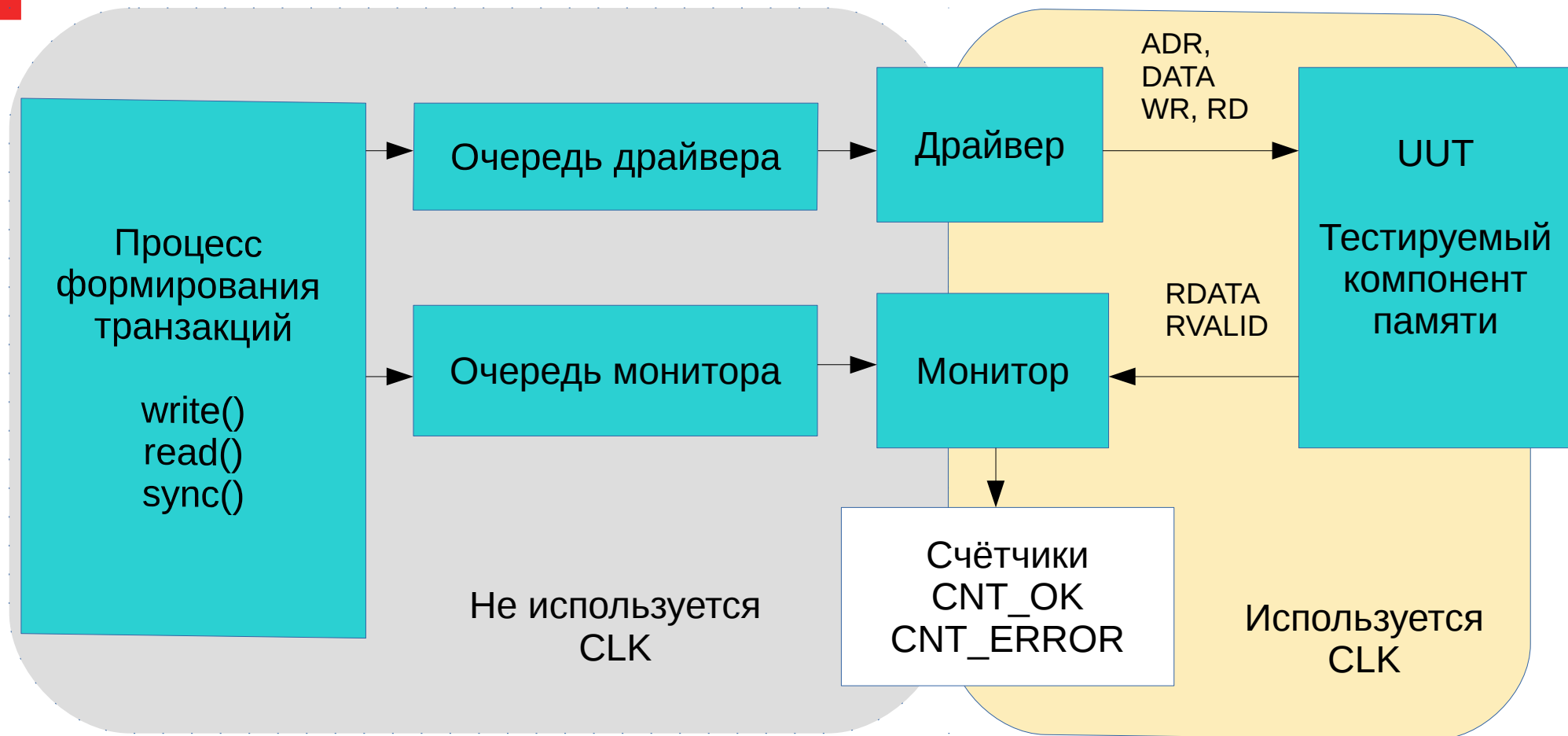
## Пример 2.1.4.1



# Основные цели

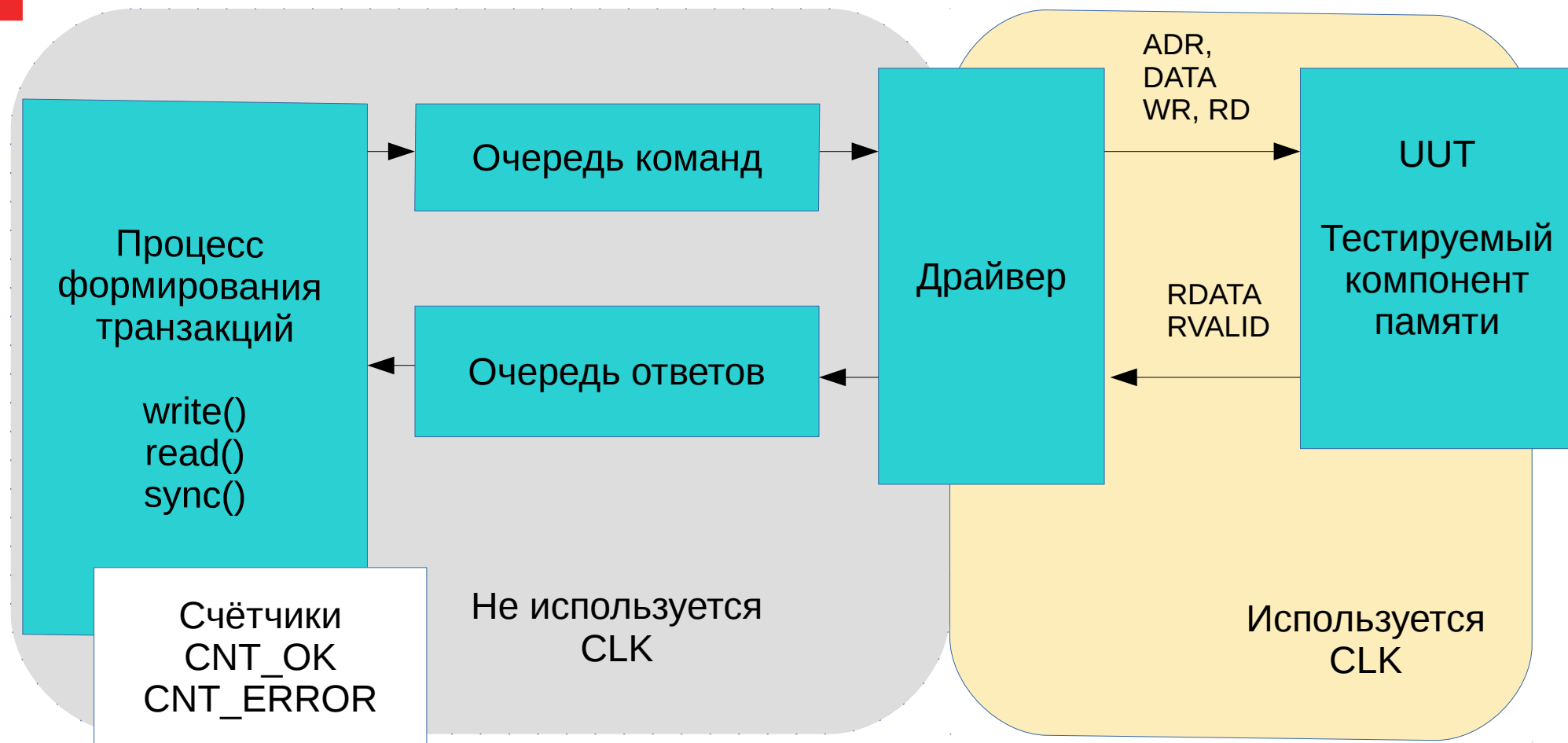
- Проверка многопортовой памяти в различных режимах работы
- Применение транзакций в системе тестирования

# Структура стенда тестирования



- Драйвер — преобразует транзакции в сигналы
- Монитор — сравнивает ответ с ожидаемым значением

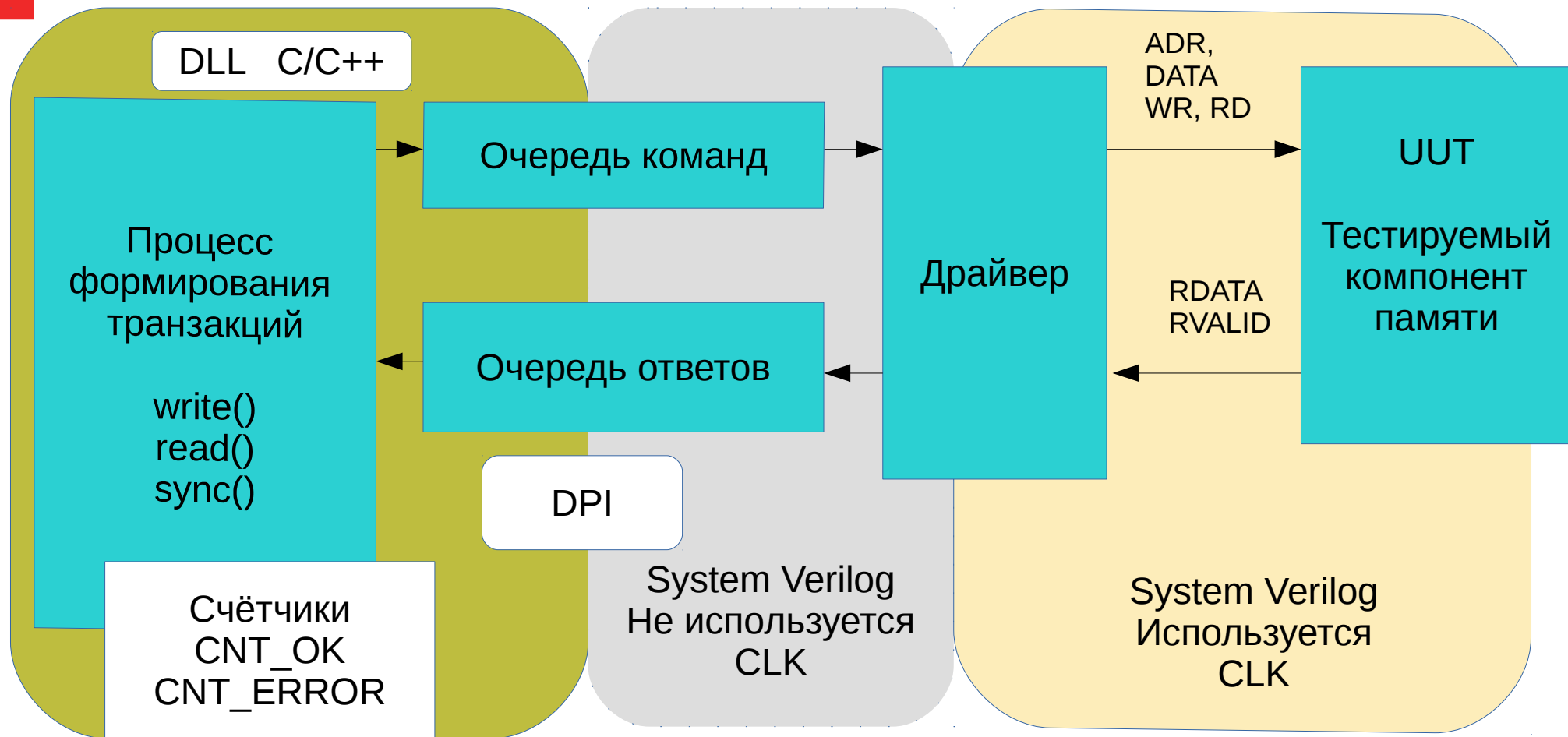
## Второй вариант



Доступны все возможности языка высокого уровня для формирования циклов, ветвлений, вызовов подпрограмм.

Это аналог программы на C/C++

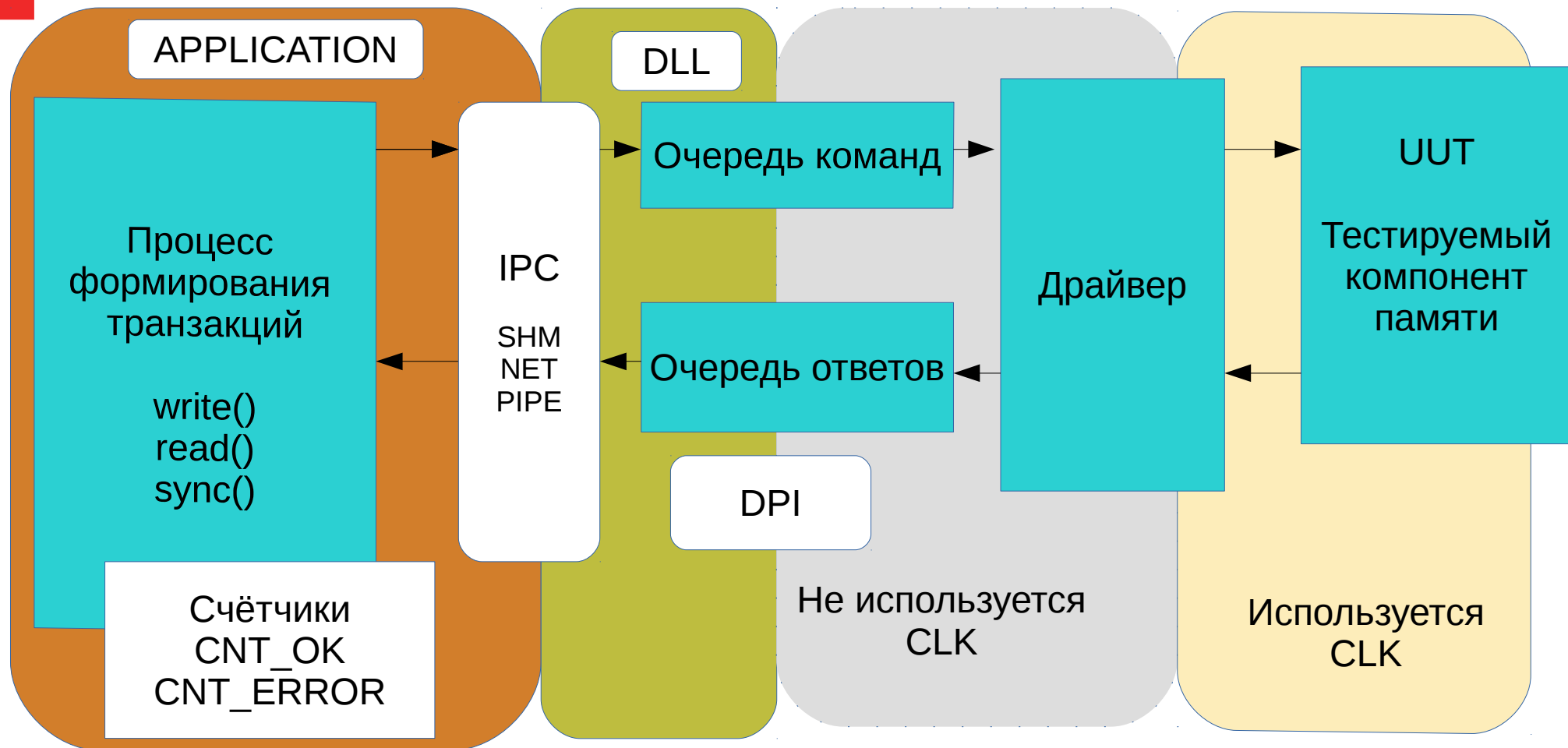
# Третий вариант — подключение DLL



К симулятору подключается DLL через интерфейс DPI.

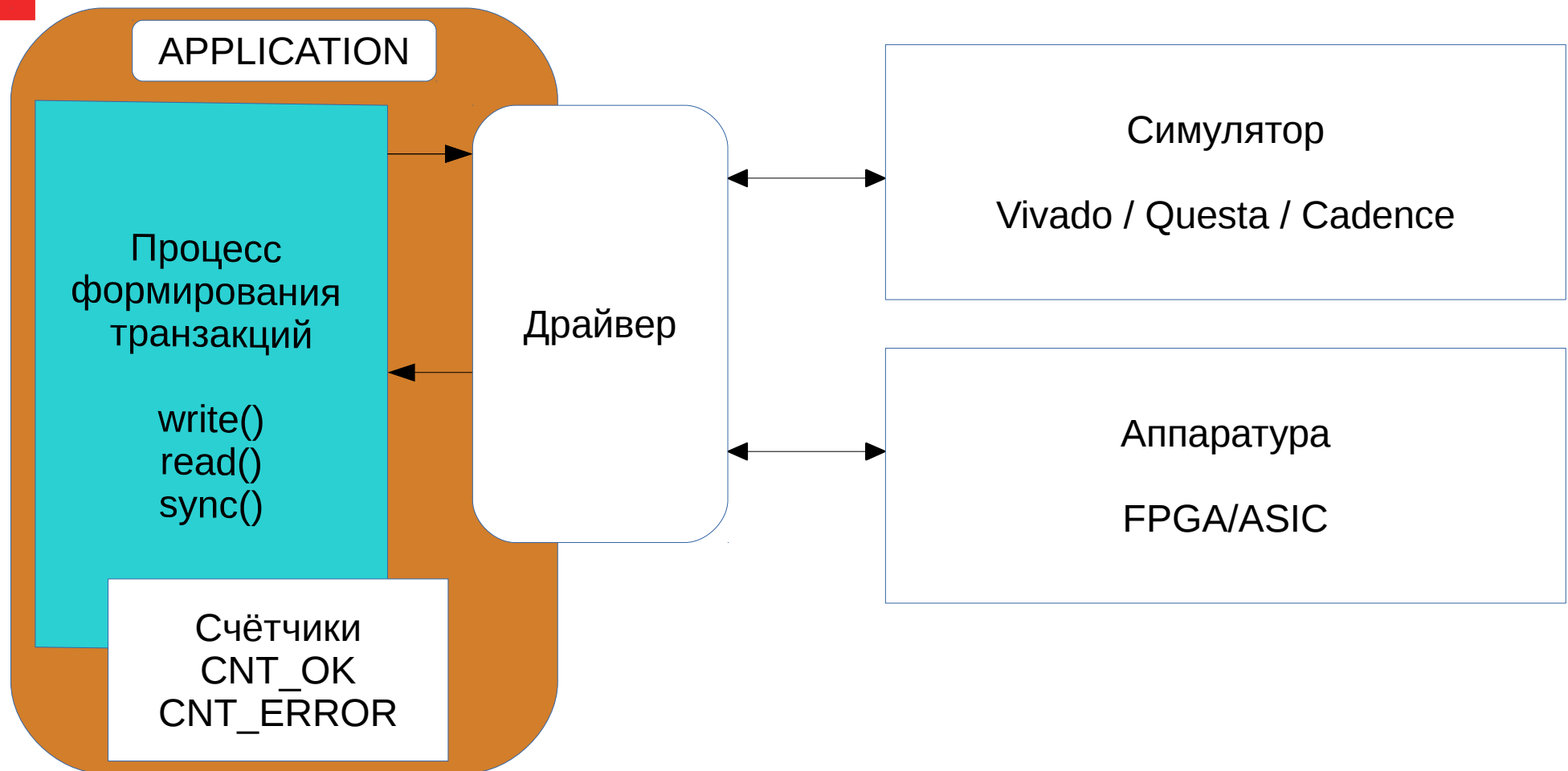
Функции из DLL могут быть вызваны из кода System Verilog

# Взаимодействие с внешней программой



APPLICATION — произвольное приложение, взаимодействует с симулятором через какой-то механизм IPC (inter process communication)

# Выполнение на аппаратуре и на симуляторе



APPLICATION — может использовать различные драйверы для выполнения кода на симуляторе или на аппаратуре

Xilinx Vitis — режимы Emulation-SW, Emulation-HW, Hardware

# Test framework

Универсальные:

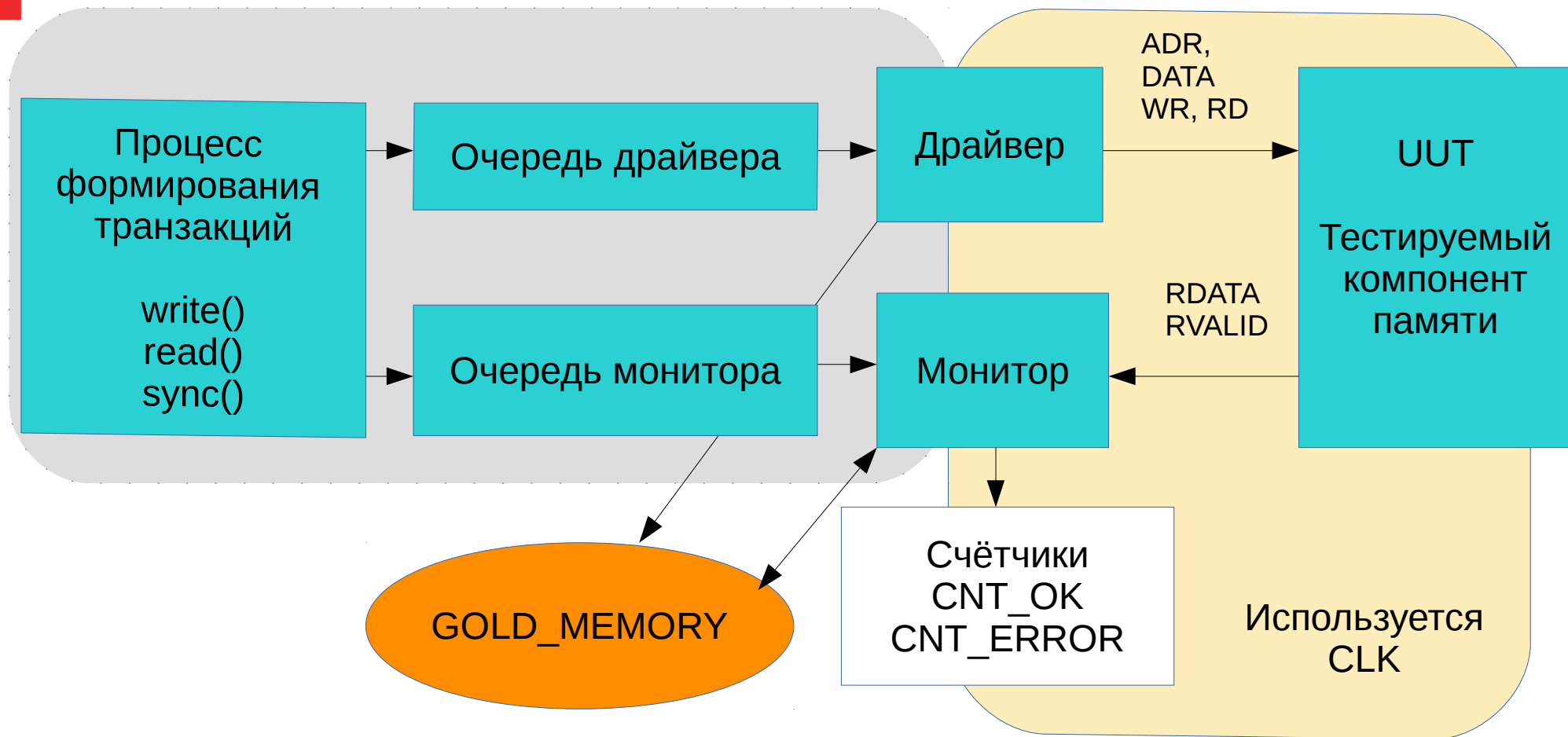
- UVM
- UVVM
- OSVVM
- Cocotb

Всегда есть возможность разработать свой собственный

Цель данного примера: показать как разработать свою систему тестирования



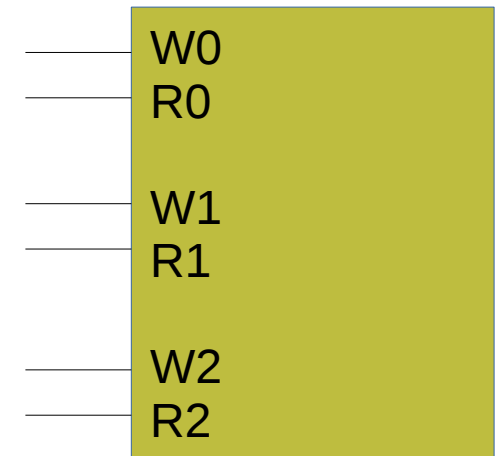
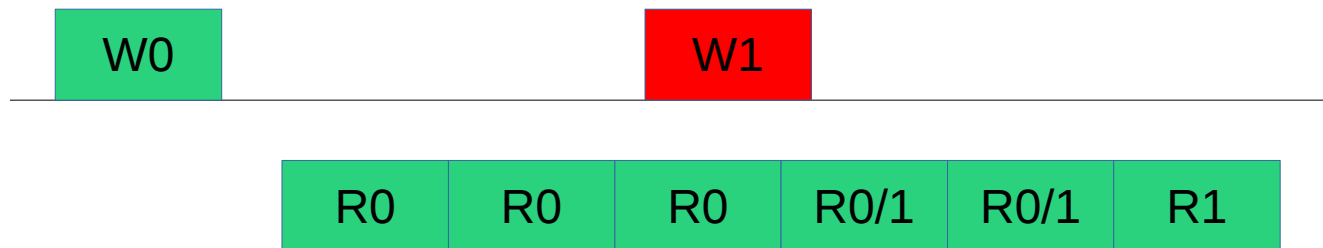
# Структура стенда тестирования



- GOLD\_MEMORY — эталонная память

# Неопределённость чтения

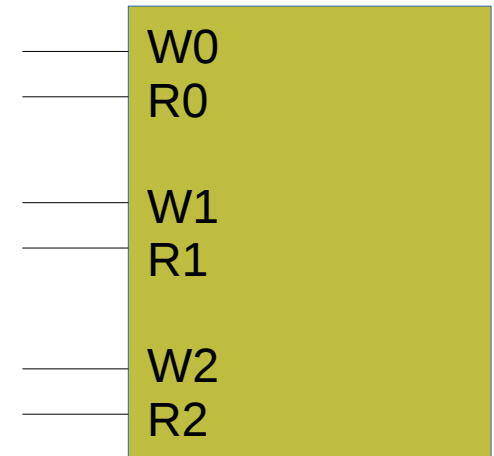
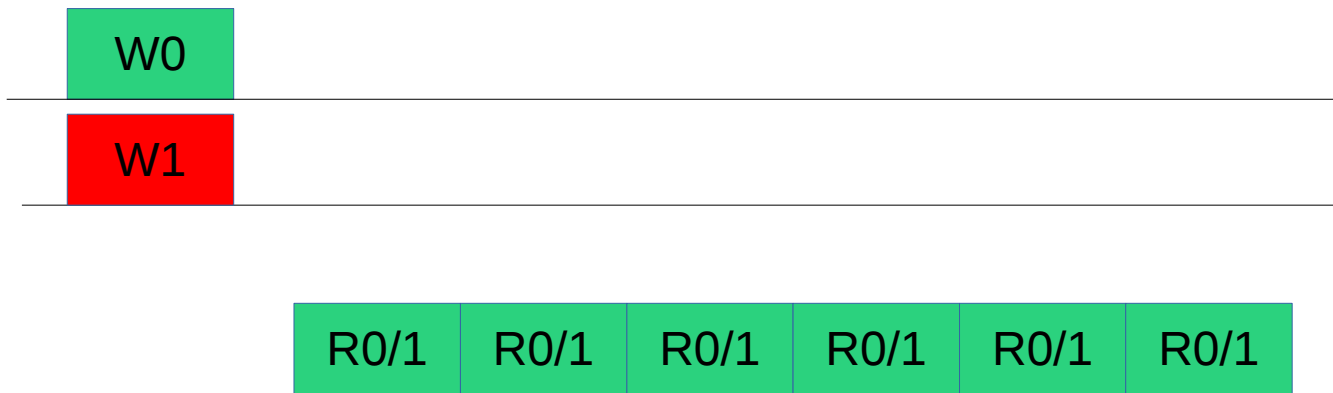
Компонент имеет несколько портов памяти  
Неизвестно, на каком такте появится новое значение



- Номер такта с которого начнут поступать новые данные зависит от загруженности других портов и не может быть предсказан
- Монитор должен учитывать эту возможность и считать оба результата правильными

# Неопределённость записи

Компонент имеет несколько портов памяти  
Неизвестно, какое значение будет записано  
при одновременной записи на нескольких портах



- Арбитр выполнит обе операции записи, невозможно предсказать порядок выполнения операций
- Монитор должен учитывать эту возможность и считать оба результата правильными

# GOLDEN\_MEMORY

Компонент для каждого порта хранит данные:

- Текущее записанное значение
- Предыдущее записанное значение
- Номер такта на котором была запись

|                        |
|------------------------|
| DATA_CURR<br>TICK_CURR |
| DATA_LAST<br>TICK_LAST |

- При запросе эталонного значения проверяется номер такта записи по всем портам
- Возвращается до шести возможных значений
- Монитор должен проверить принятое значения на одно из ВОЗМОЖНЫХ
- Возможно формирование событий при наступлении коллизий