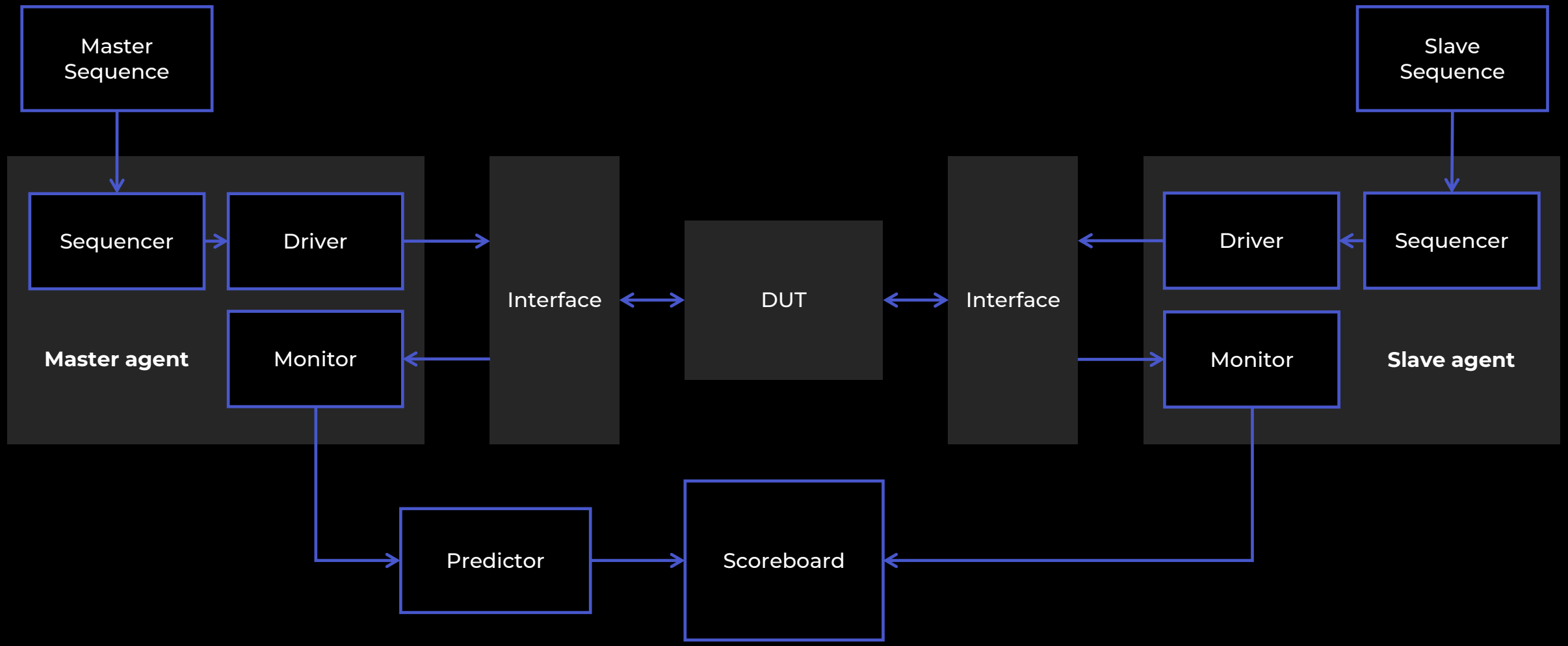
БЮРО 1440

Генерация преднамеренных ошибок в UVM тесте

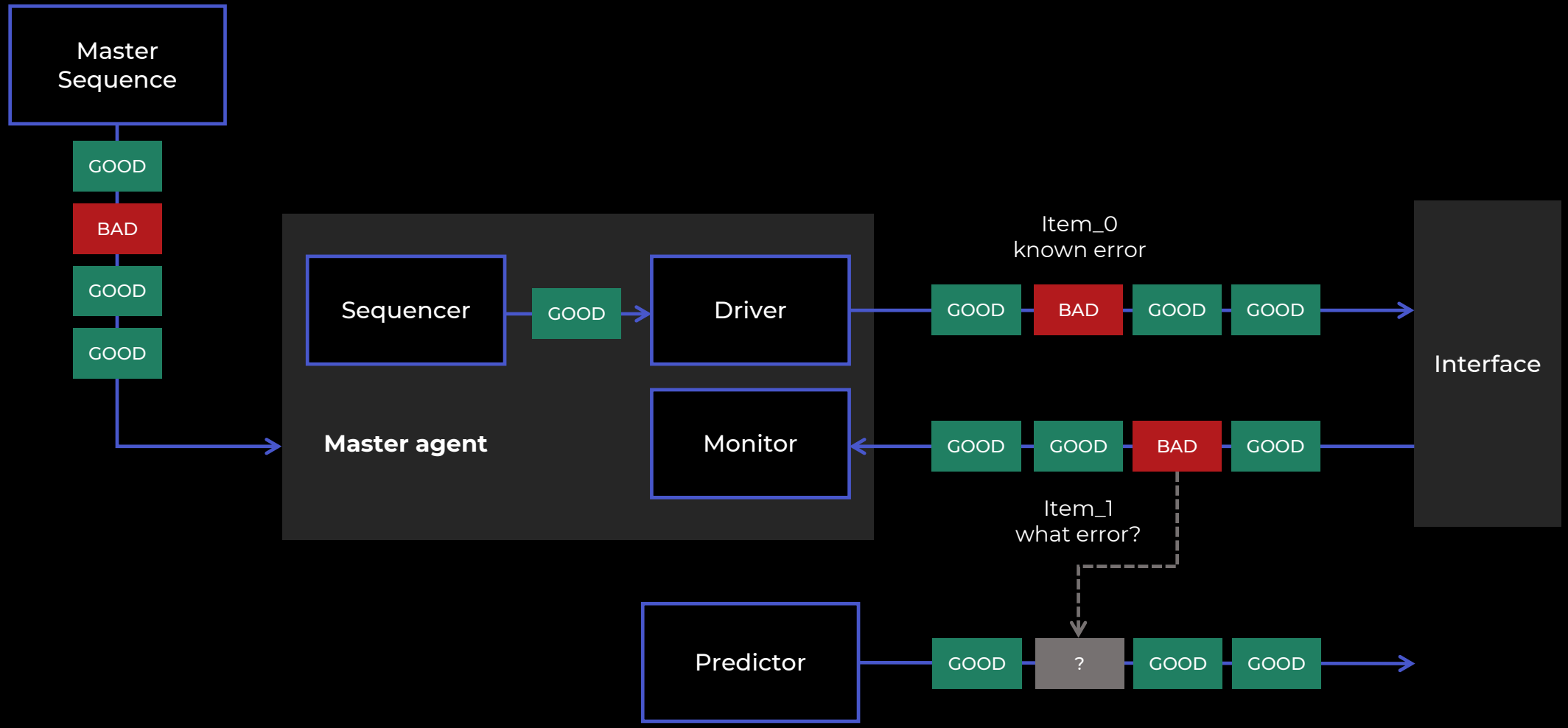
Спикер

Андрей Ефимов
Ведущий инженер-верификатор
БЮРО 1440

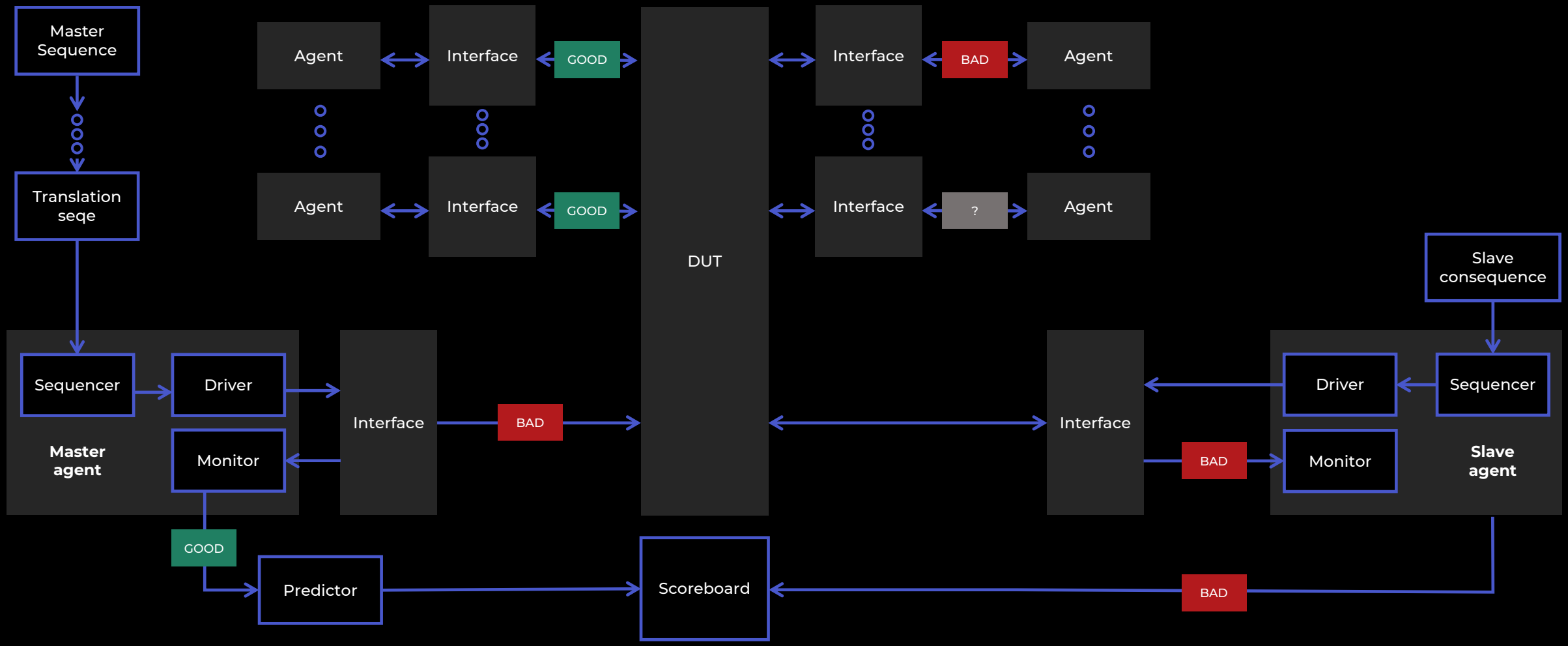
Традиционное представление тестового стенда из книг



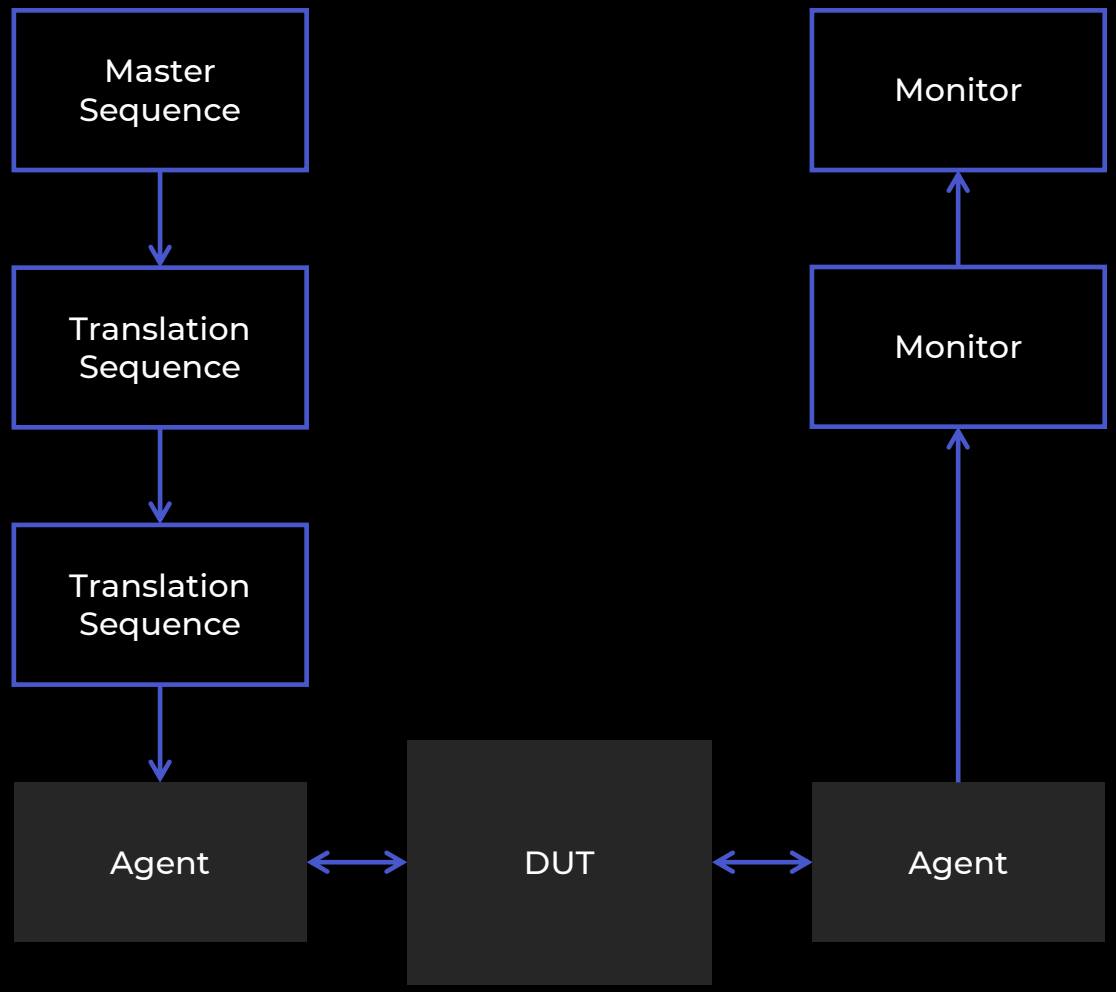
Традиционный пример следования ошибок

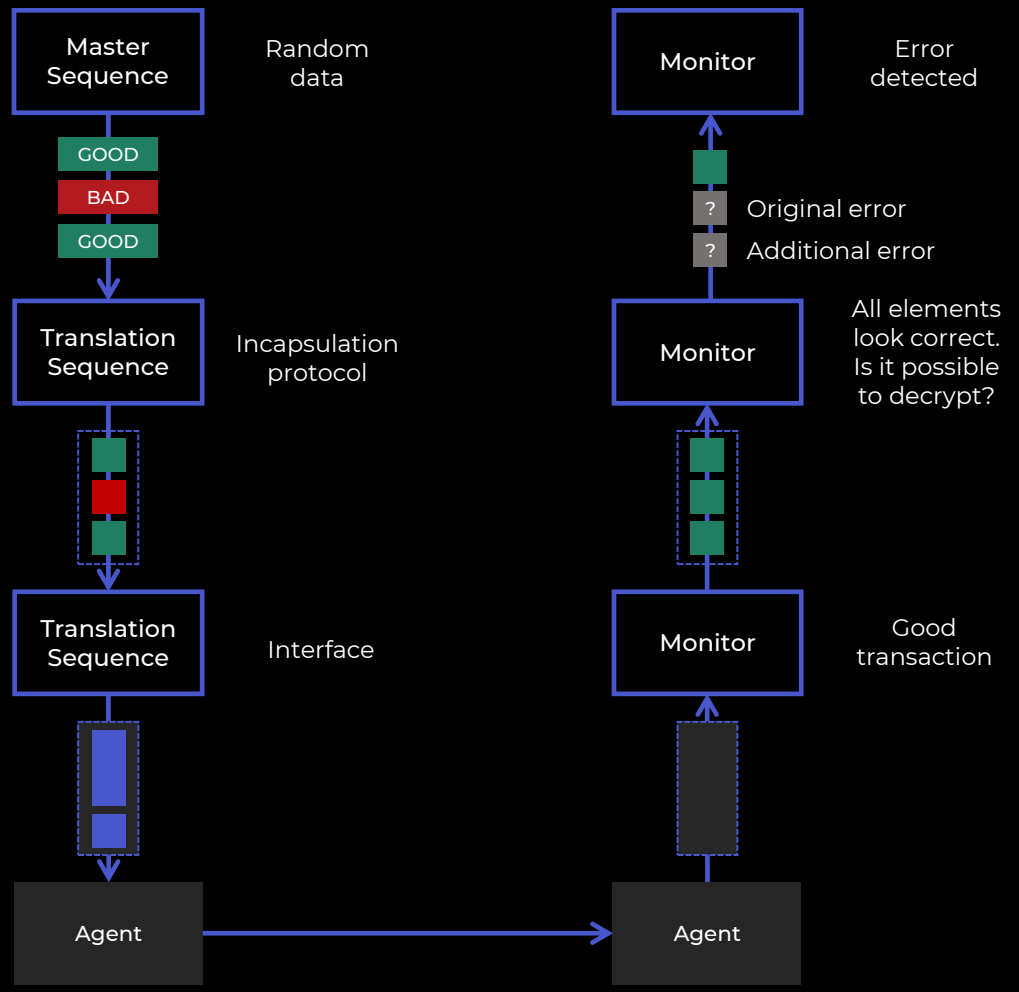


Реалистичный вид тестового стенда



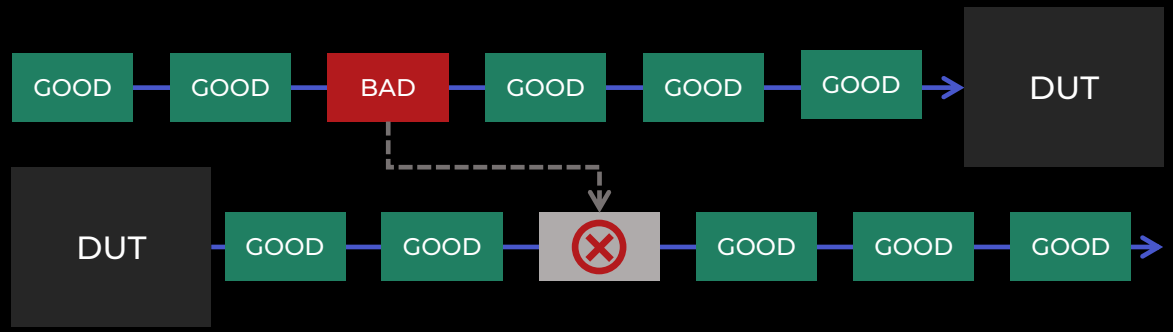
Ошибки в иерархических протоколах



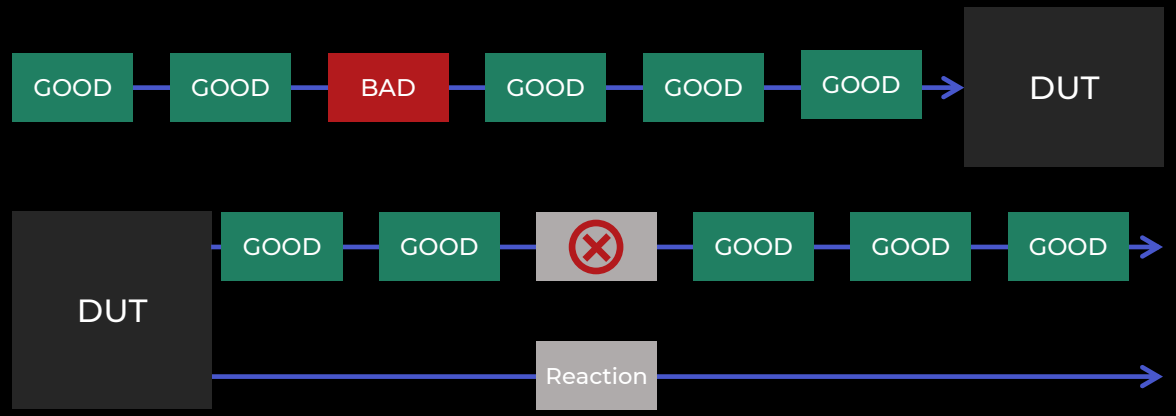


Возможные реакции на ошибку

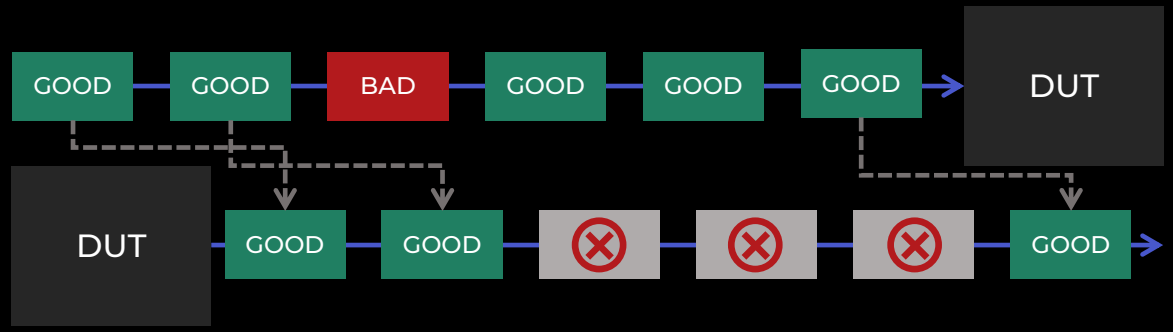
Реакция «1 к 1»



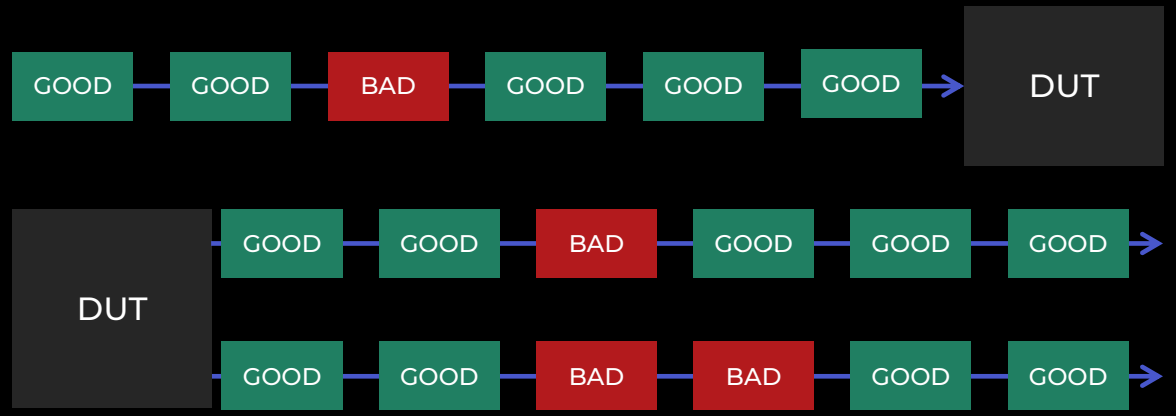
Реакция с индикацией статуса или прерывания



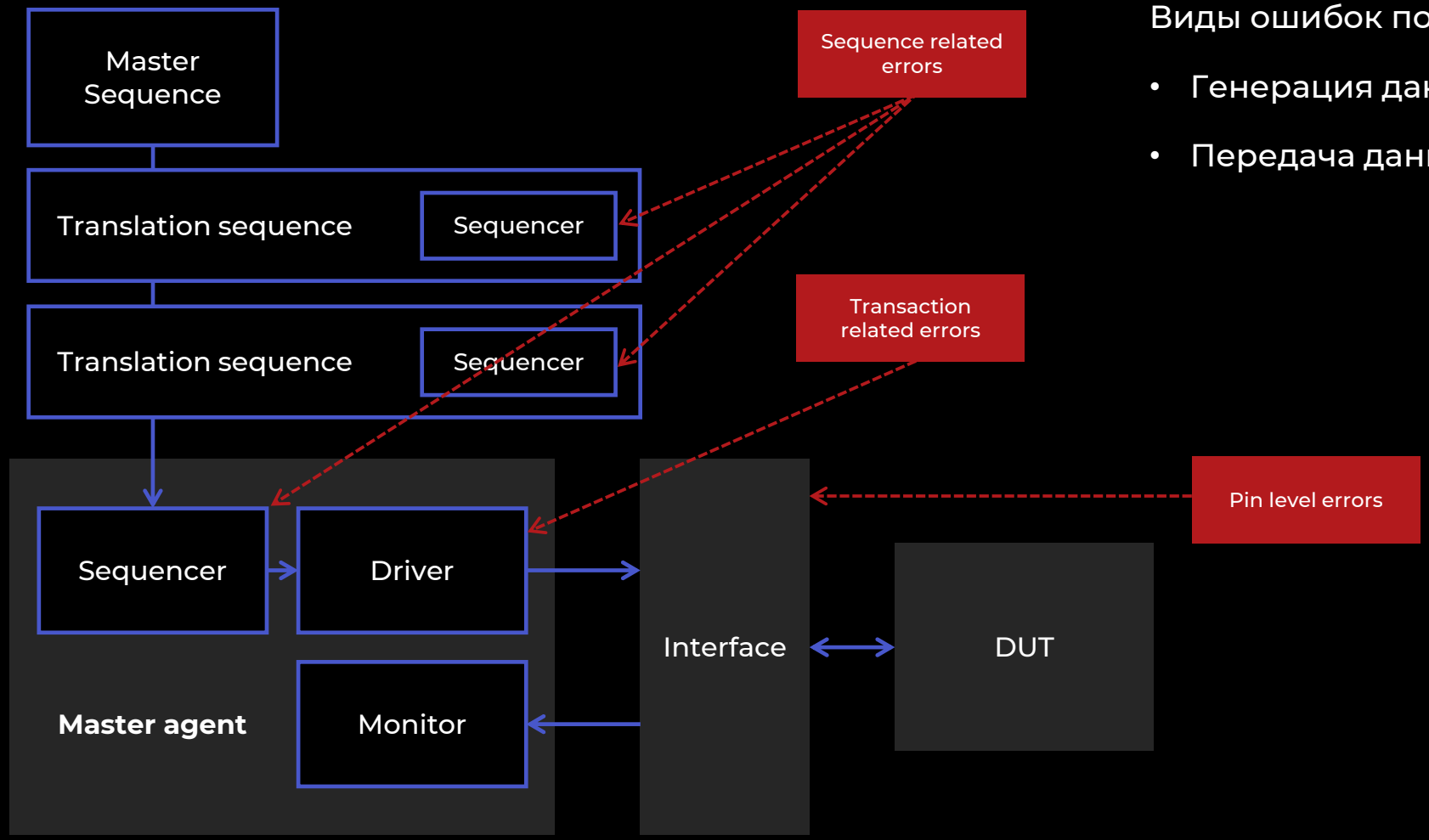
Реакция «Каскад ошибок»



Реакция с ошибкой по нескольким интерфейсам



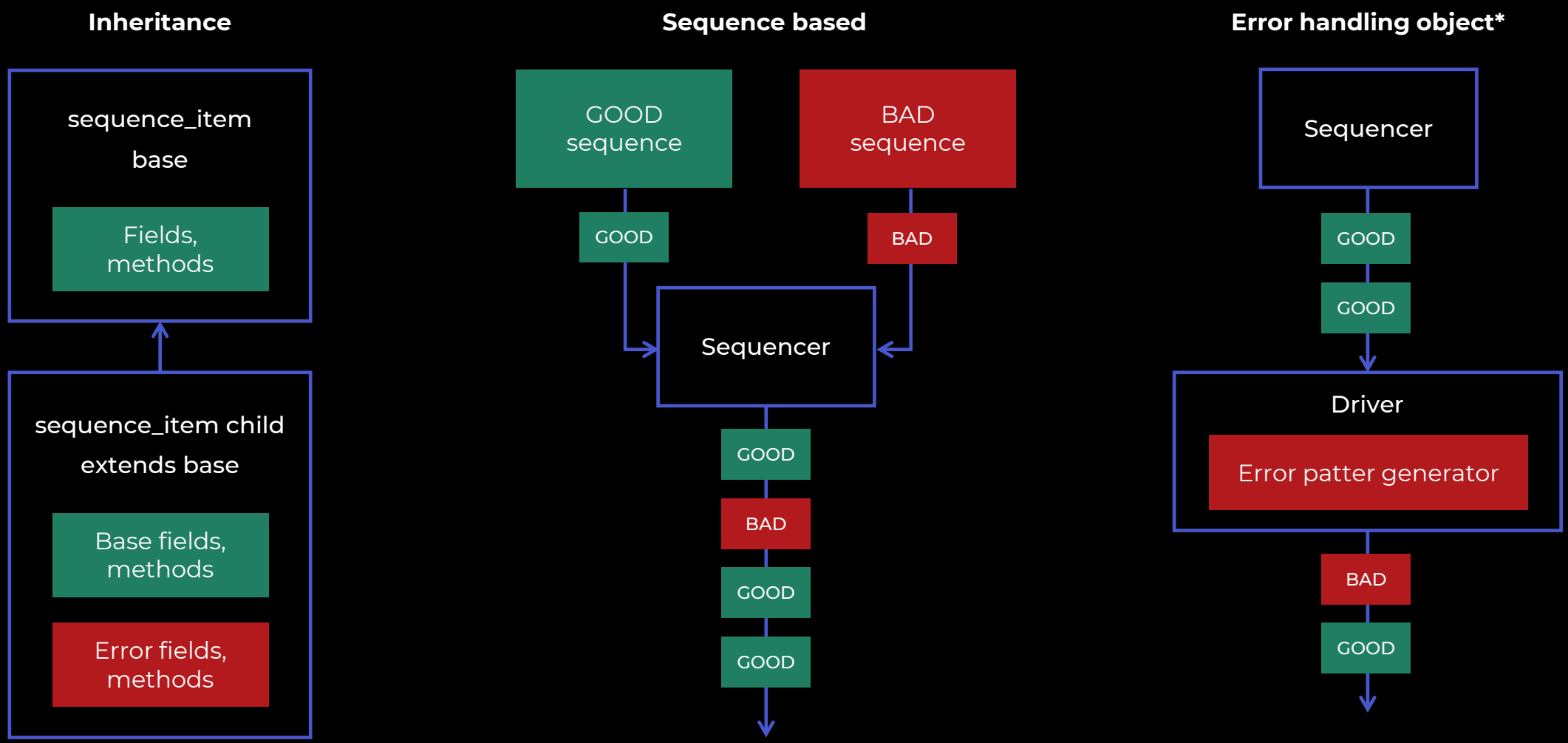
Этапы добавления ошибок



Виды ошибок по месту возникновения:

- Генерация данных (sequence)
- Передача данных (transaction / driver, pin level)

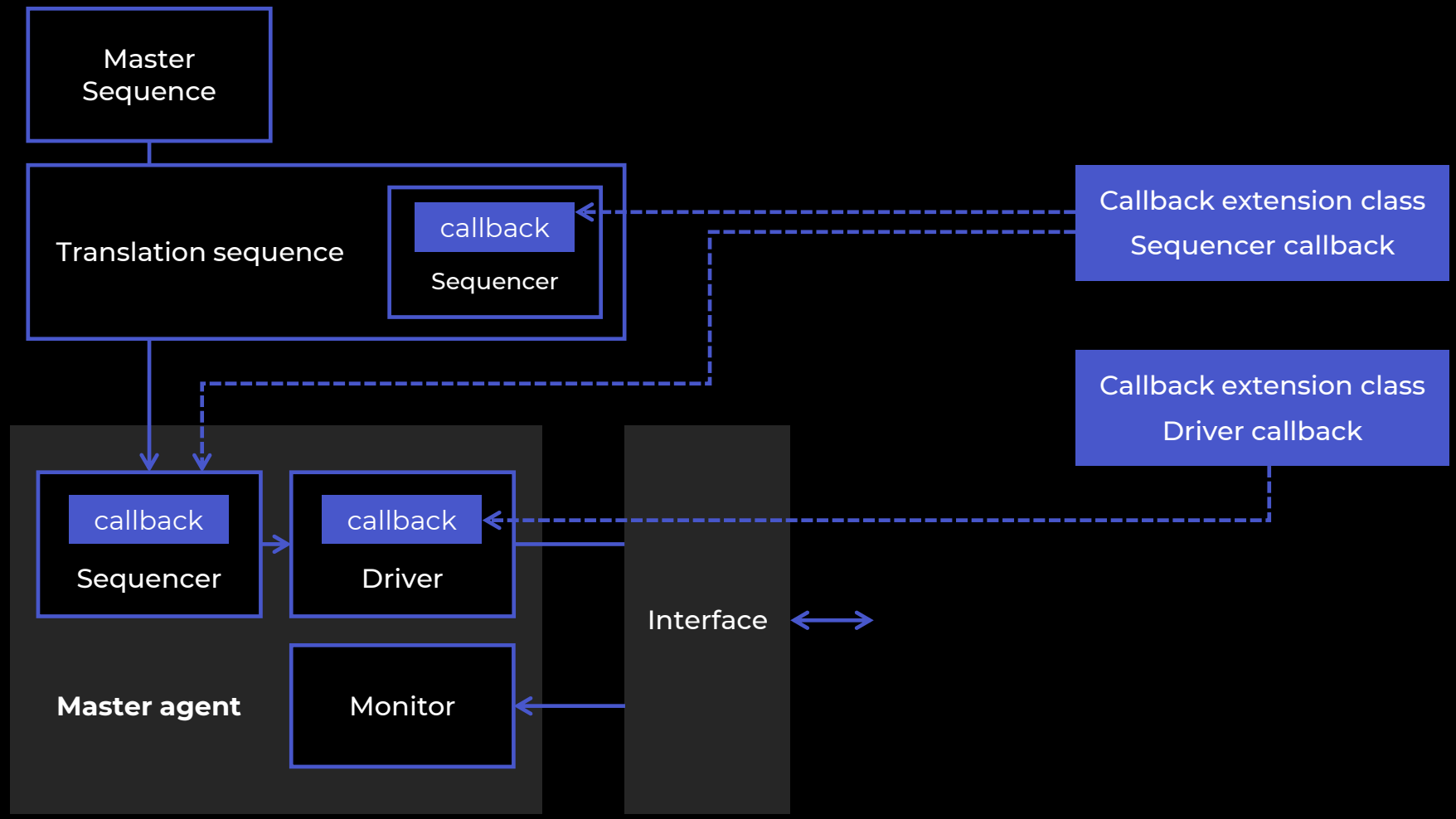
Традиционный пример следования ошибок



* Kurt Schwartz, Tim Corcoran "Error Injection: When Good Input Goes Bad"

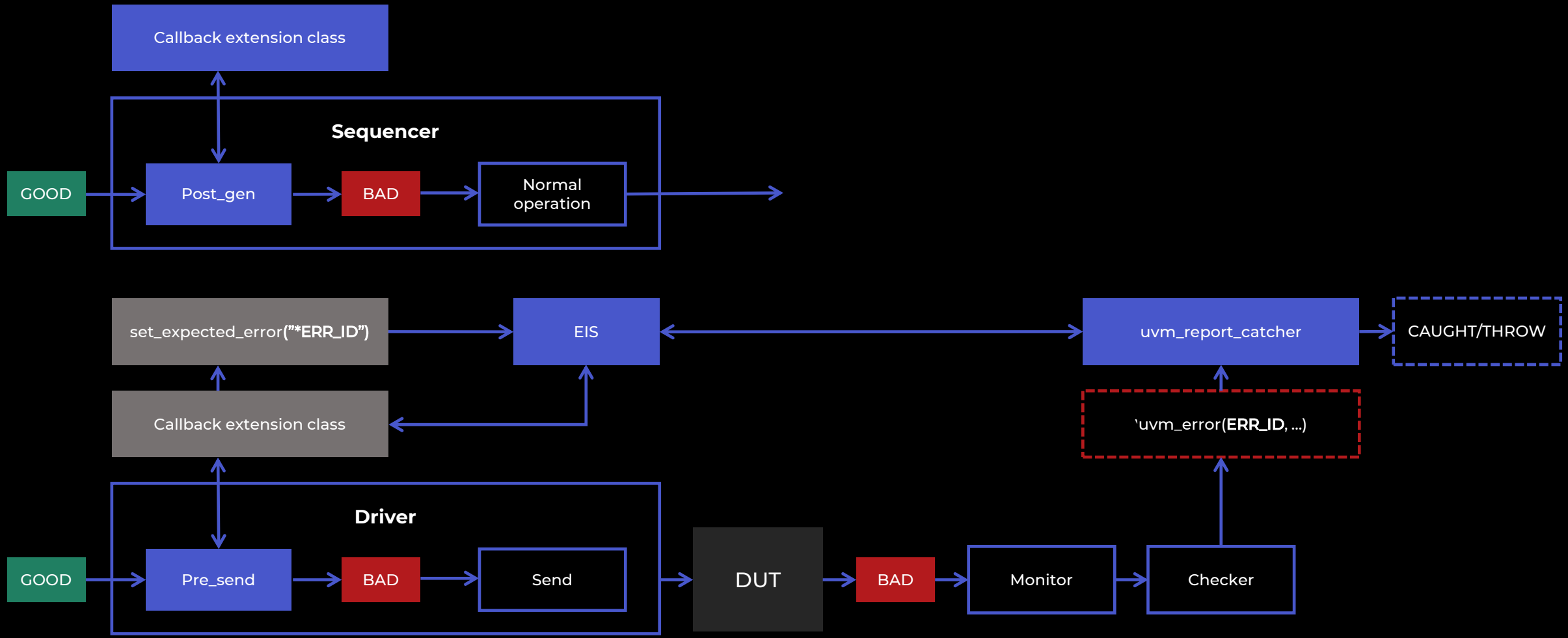
Способы добавления ошибок

Error injection server*

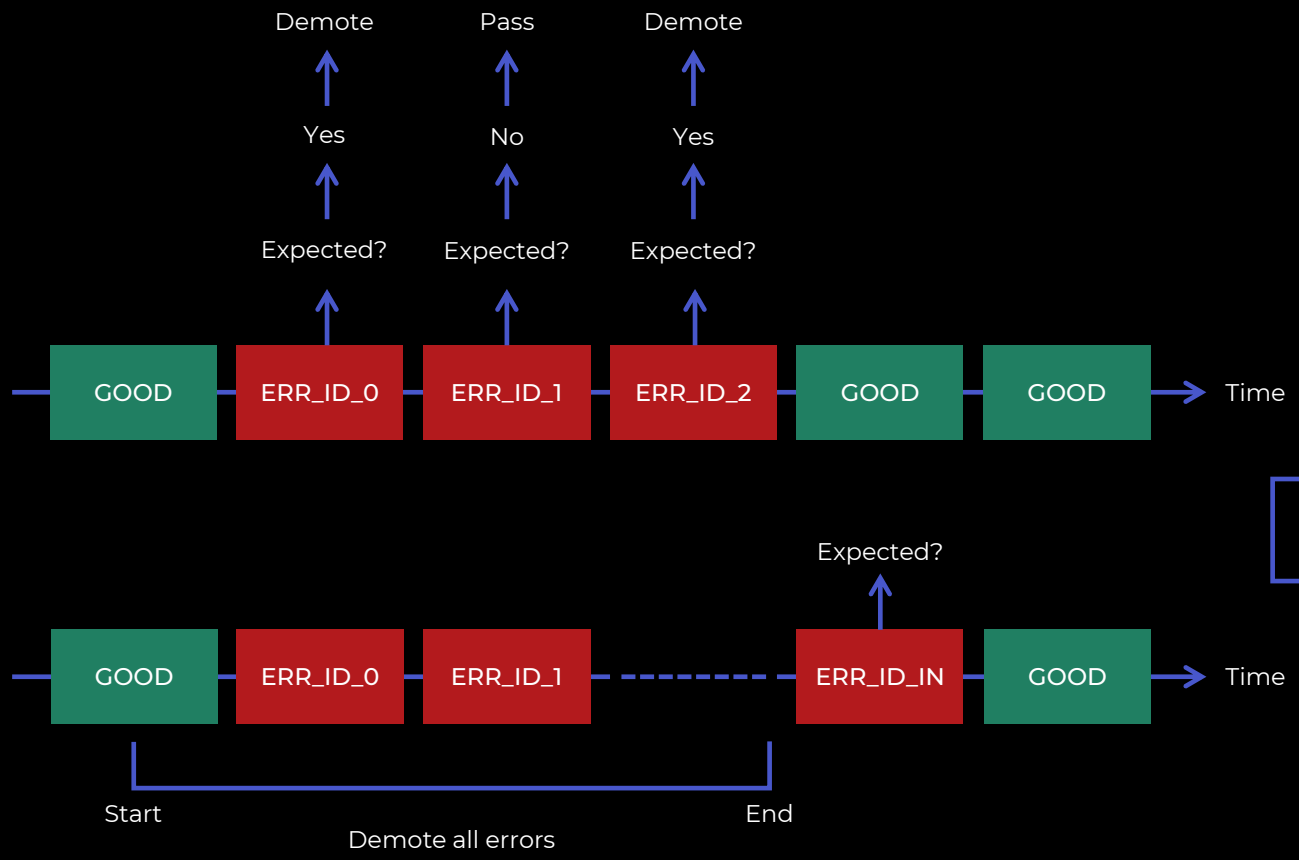


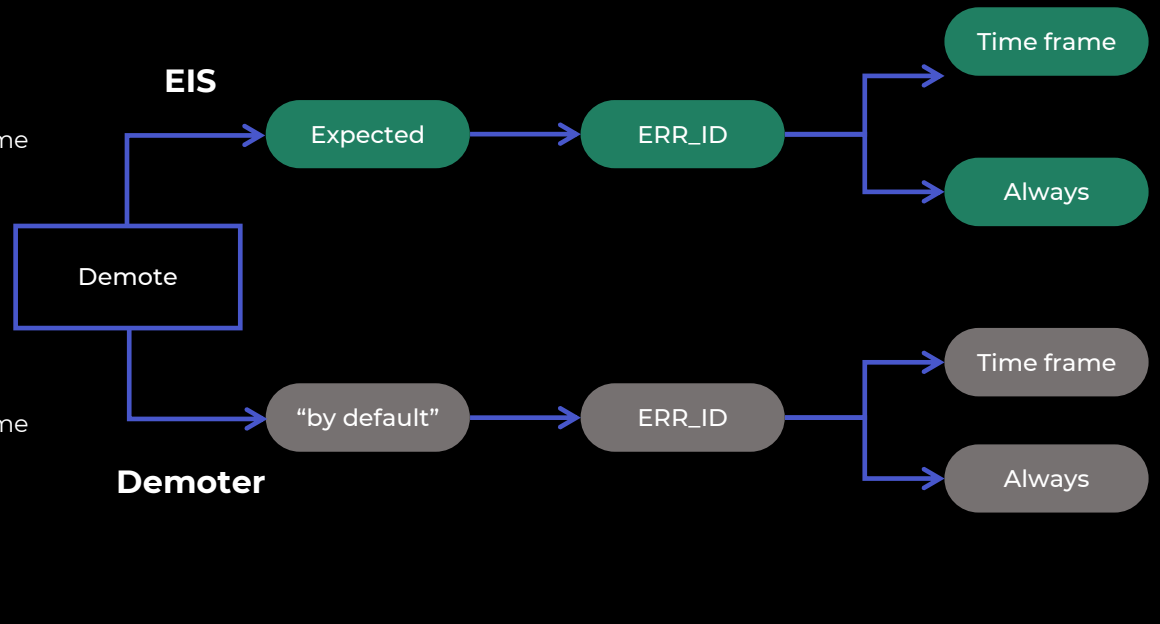
* Jeremy Ridgeway, Hoe Nguyen “Error Injection in a Subsystem Level Constrained Random UVM Testbench”

Способы добавления ошибок



Требования к перехвату ошибок





Error injection server

Позволяет использовать «стандартные» компоненты без дополнительных модификаций

Единый механизм для:

Добавление ошибок

Оценки ошибок
и ведения статистики

Имитация действий внешней системы при возникновении ошибки (опционально)

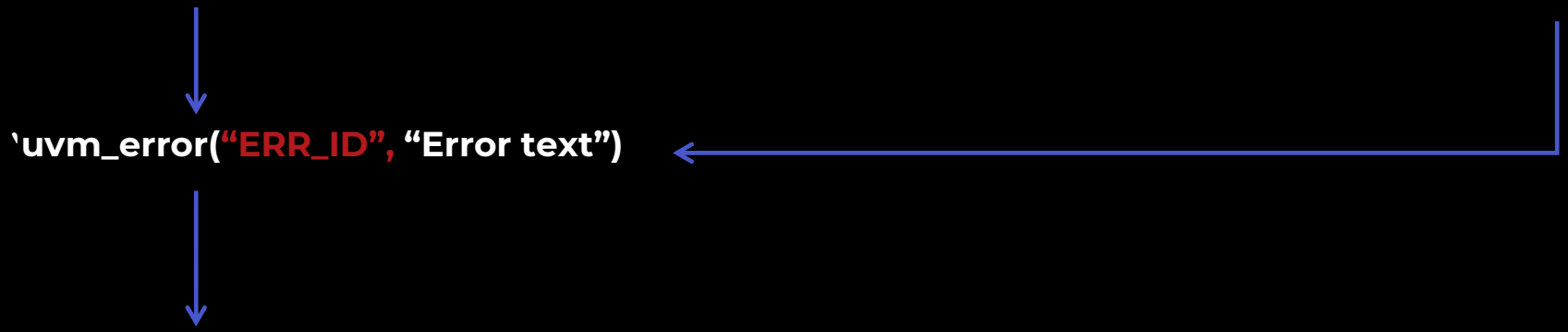
Структура сообщения об ошибке

Покупное IP (VIP, UVC)

ERR_ID = ERR_TYPE

Компонент собственной разработки (VIP, UVC)

ERR_ID = {COMPONENT_NAME, ERR_TYPE}

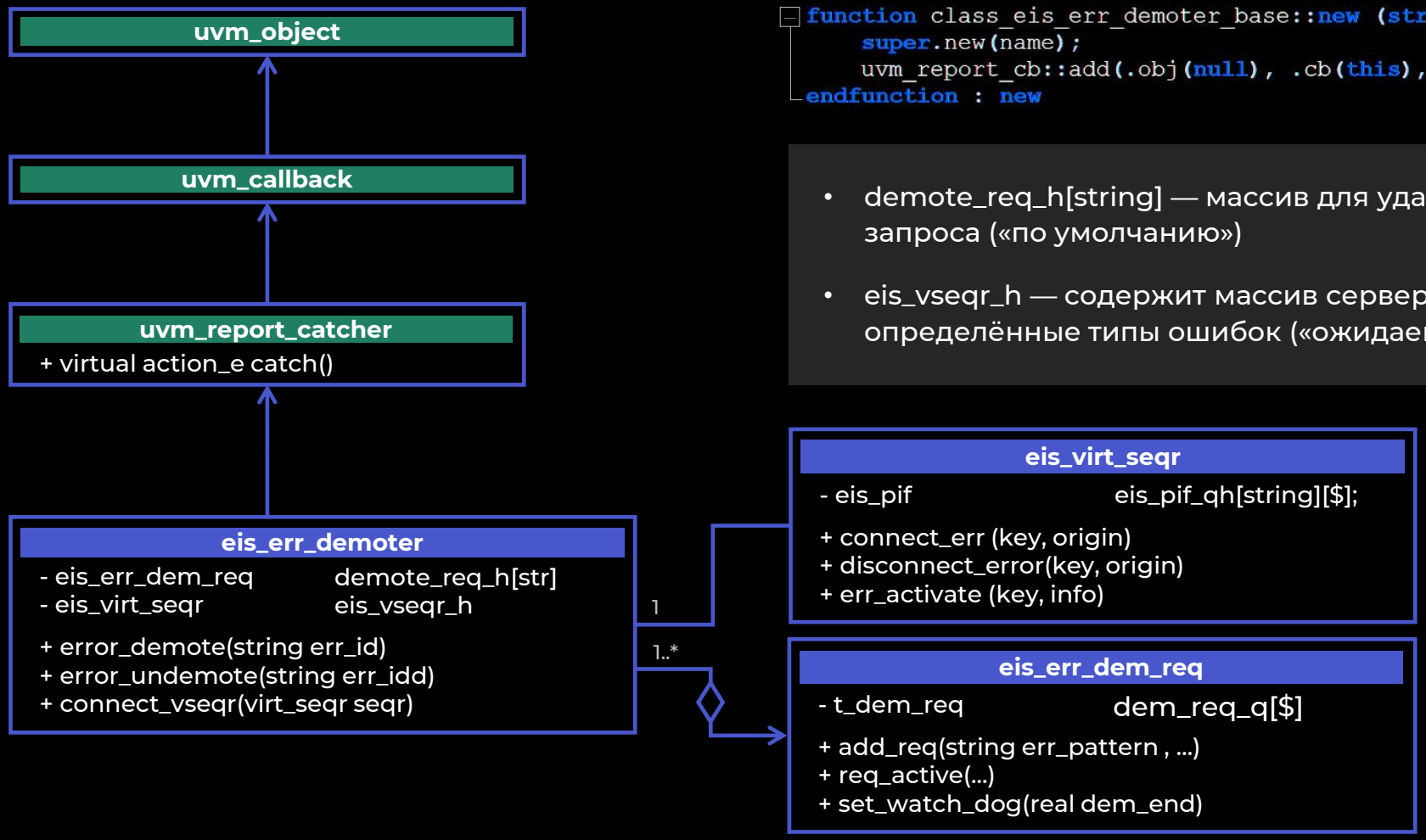


UVM_ERROR my_components.sv(0) @ 0.0ns: tb_top.env_h.my_inst [ERR_ID] Error text

Тип или серьезность ошибки (fatal, error, warning).
На этапе добавления ошибки мы уже знаем насколько серьезный сбой последует при обработке преднамеренной ошибки

- **Время ошибки**
- **Путь до источника ошибки**
- **ID ошибки**
- Текст ошибки

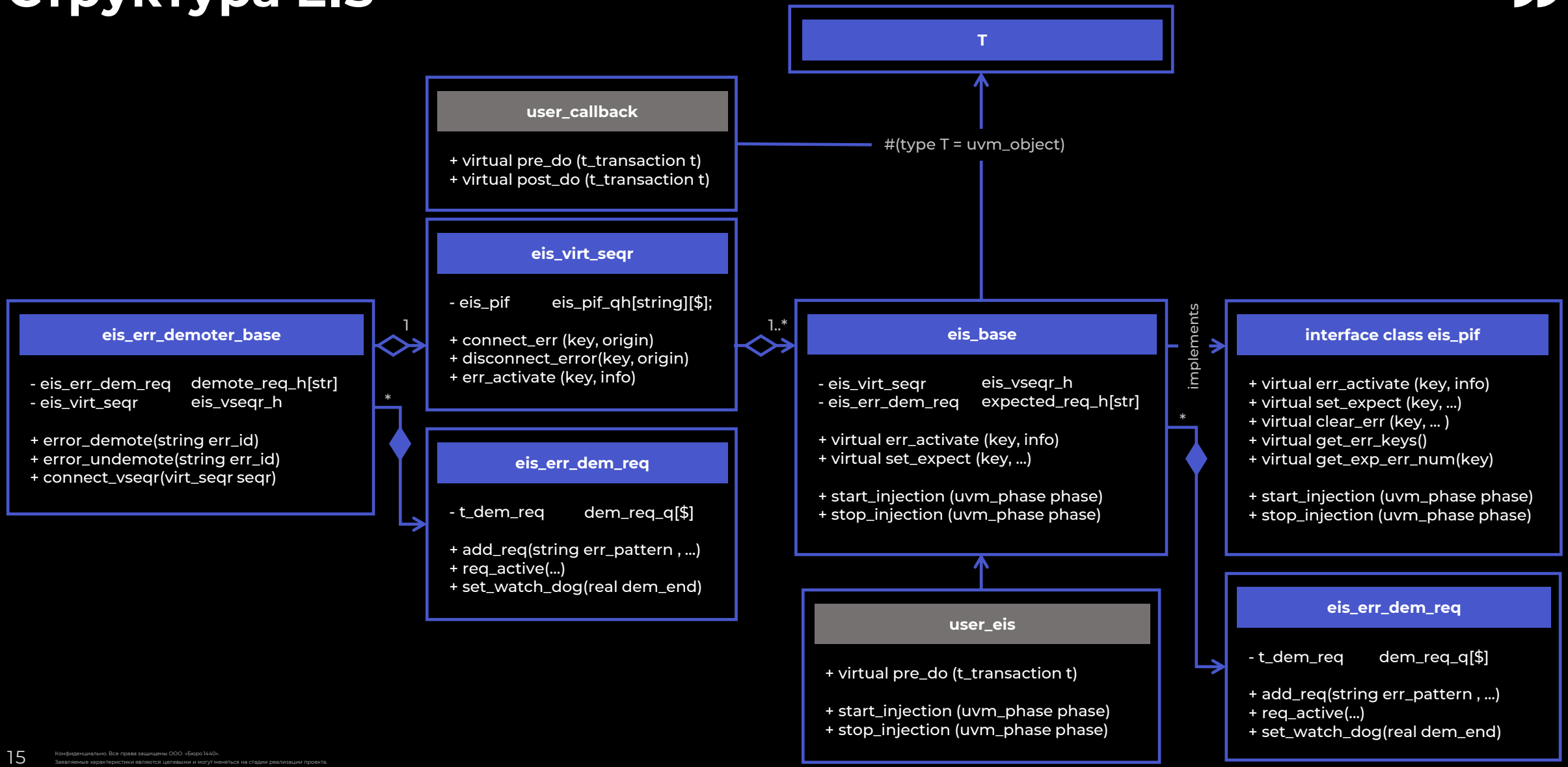
UVM report catcher




```
function class_eis_err_demoter_base::new (string name = "class_eis_err_demoter_base");
    super.new(name);
    uvm_report_cb::add(.obj(null), .cb(this), .ordering(UVM_PREPEND));
endfunction : new
```

- demote_req_h[string] — массив для удаления ошибок без запроса («по умолчанию»)
- eis_vseqr_h — содержит массив серверов, ожидающих определённые типы ошибок («ожидаемые» ошибки)

Структура EIS



Конфигурация EIS

```
typedef struct {
    string error_id     ; // same as error_active(err_key,...)
    string rpt_context  ; // test bench path
    string rpt_message  ; // full reported message
    int    new_severity ; // what to change severity to (default: UVM_INFO)
    int    new_verbosity; // what to change verbosity to (default: UVM_HIGH)
} t_eis_err_info;

typedef struct {
    string err_type         = "ERR_NONE"; // Error type (ERR_ID), "ERR_NONE" - special type which means "no error"
    int    err_weight       = 1;          // Similar to the constraint "dist :="
    int    err_num_gen      = 1;          //     How many of this type will be generated
                                          //         during the current test case (start_inject)
    int    err_in_trans_max = 1;          //     How many errors of this type can be
                                          //         inserted into one transaction
} t_err_set;


typedef struct {
    t_err_set       err_set[$]  ; // Set of all possible types of errors
    int             err_num_max ; // Maximum number of all errors in one transaction
} t_eis_cfg;

// Which types of errors can be generated in the same  transaction
// (mutually exclusive errors or not)
typedef bit t_eis_err_mtx[string][string];

typedef struct {
    uvm_severity  demote_to_severity;
    uvm_verbosity demote_to_verbosity;
    int           expected_count;
    real          dem_period[2];
    string        origin_pattern;
    ...
} t_dem_req;
```

```
// create config of the type t_eis_cfg
eis_cfg.err_set = '{
      '{cfg_h.NAME_ERR_NONE,  1,  -1, -1}
    , '{cfg_h.NAME_ERR_BIT ,  2,   3,  1}
    , '{cfg_h. ...         ,  0,  -1,  1}
    , '{...}
};
eis_cfg.err_num_max  = 2;

// set config
eis_h.set_conf(eis_cfg);

// start error injection in concrete interface driver
eis_h.start_injection(env_h.agent_interface_h.drv_h, phase);
```

Описание ошибки и реакции

```
task class_my_eis::pre_send (
      class_drv       drv_h
    , ref class_trans trans_in
);
    // randomize errors
    if (!this.randomize()) begin
        `uvm_fatal(get_name(), "randomization failed")
    end

    // inject errors
    foreach (err_type[index]) begin
        case (err_type[index])
            "ERR_NONE" : begin
                continue;
            end
            "ERR_BIT" : begin
                if (!std::randomize(err_pos_bit)) begin
                    `uvm_fatal(get_name(), "randomization failed")
                end

                // inject error into the transaction
                trans_in.set_data(err_pos_bit);

                // set the expected reaction for the error
                set_expect_error(.err_key(err_type[index]), .err_num(1), ...);
                set_expect_error(.err_key(...), .err_num(...), ...);
            end
            "ERR_DEST" : ...
            "ERR_ID"   : ...
            default : begin
                `uvm_error(get_name(), $sformatf("unavailable error code %s", err_type[index]))
            end
        endcase
    end
endtask : pre_send
```

Реализация callback метода

Рандомизация числа и типа ошибок

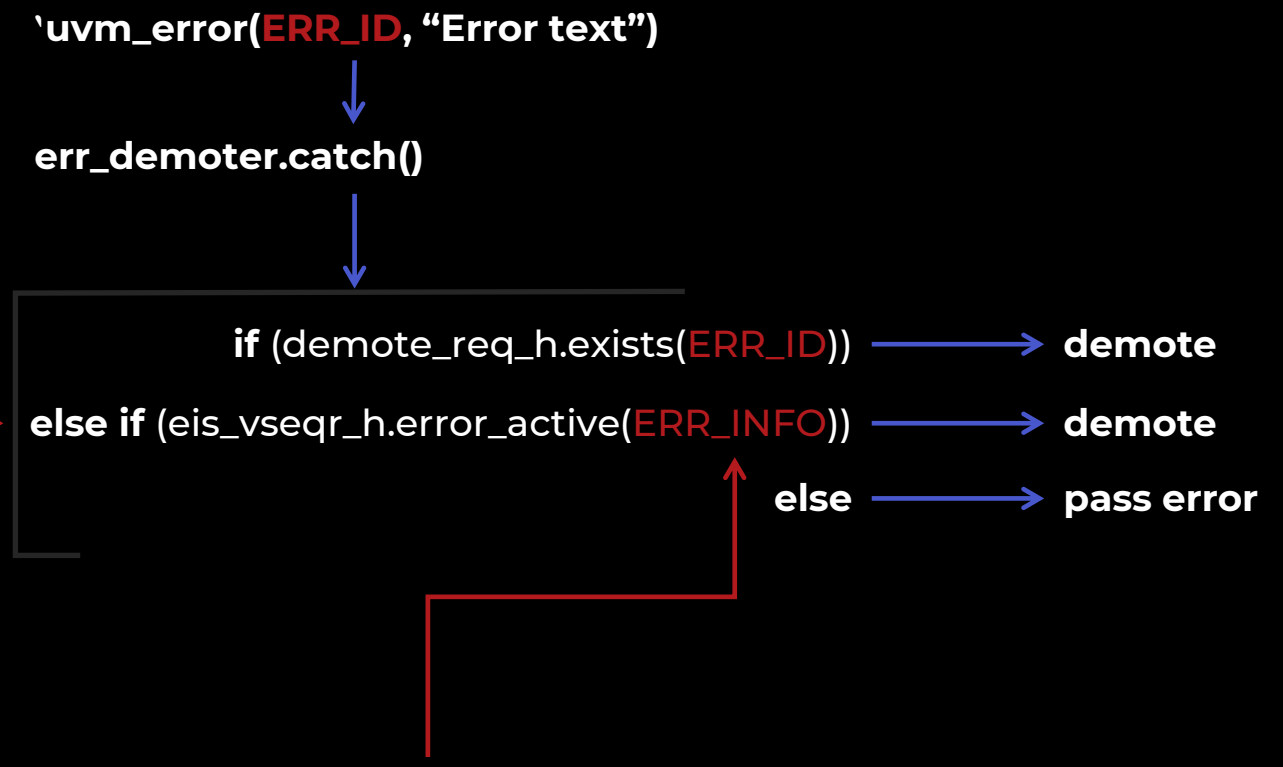
Рандомизация параметров ошибки

Создание ожидаемых реакций на ошибку

Перехват сообщения об ошибке

Ошибка независимая от места возникновения (безусловное устранение)

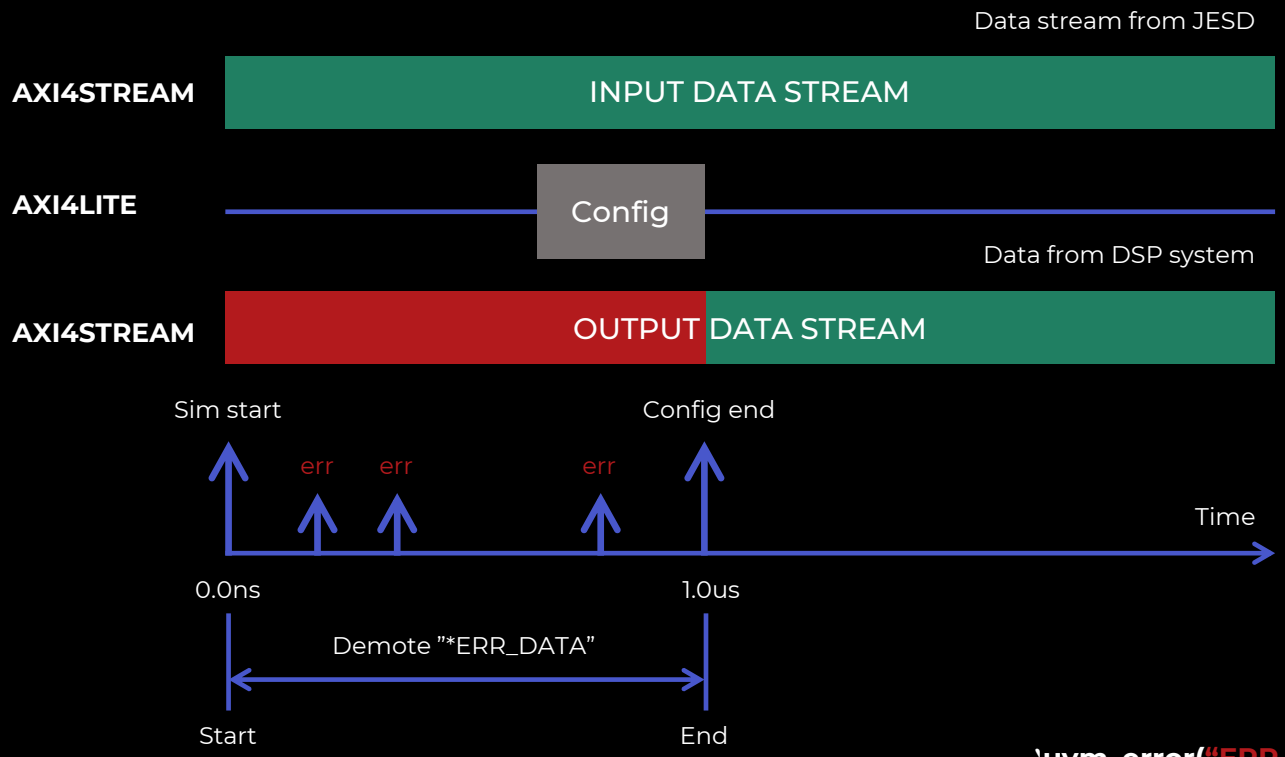
Оценка числа ошибок, места возникновения и временного окна

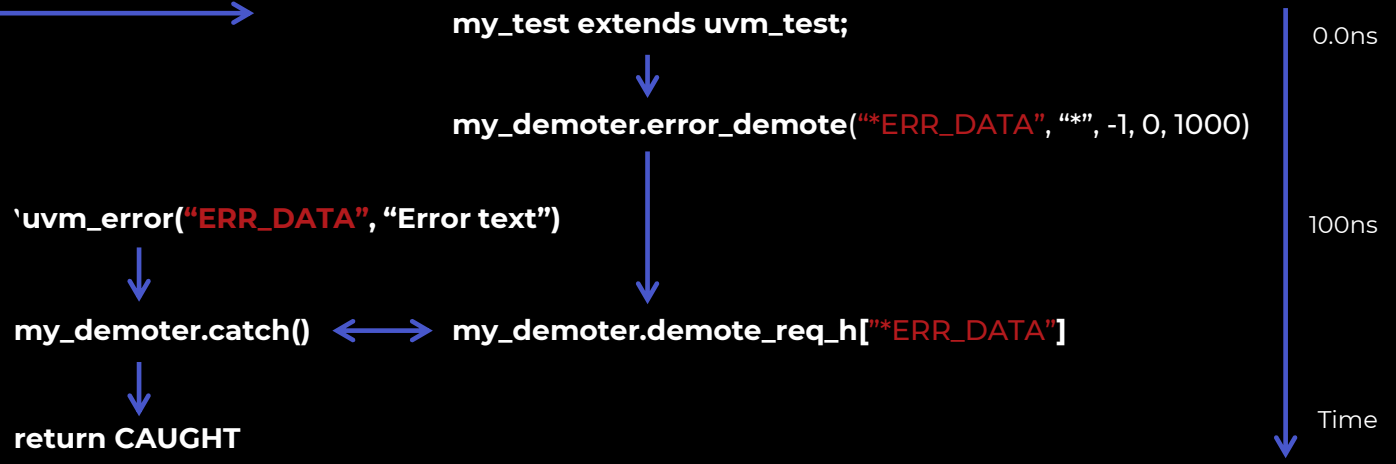



```
typedef struct {
    string error_id     ; // same as error_active(err_key,...)
    string rpt_context  ; // test bench path
    string rpt_message  ; // full reported message
    int    new_severity ; // what to change severity to (default: UVM_INFO)
    int    new_verbosity; // what to change verbosity to (default: UVM_HIGH)
} t_eis_err_info;
```

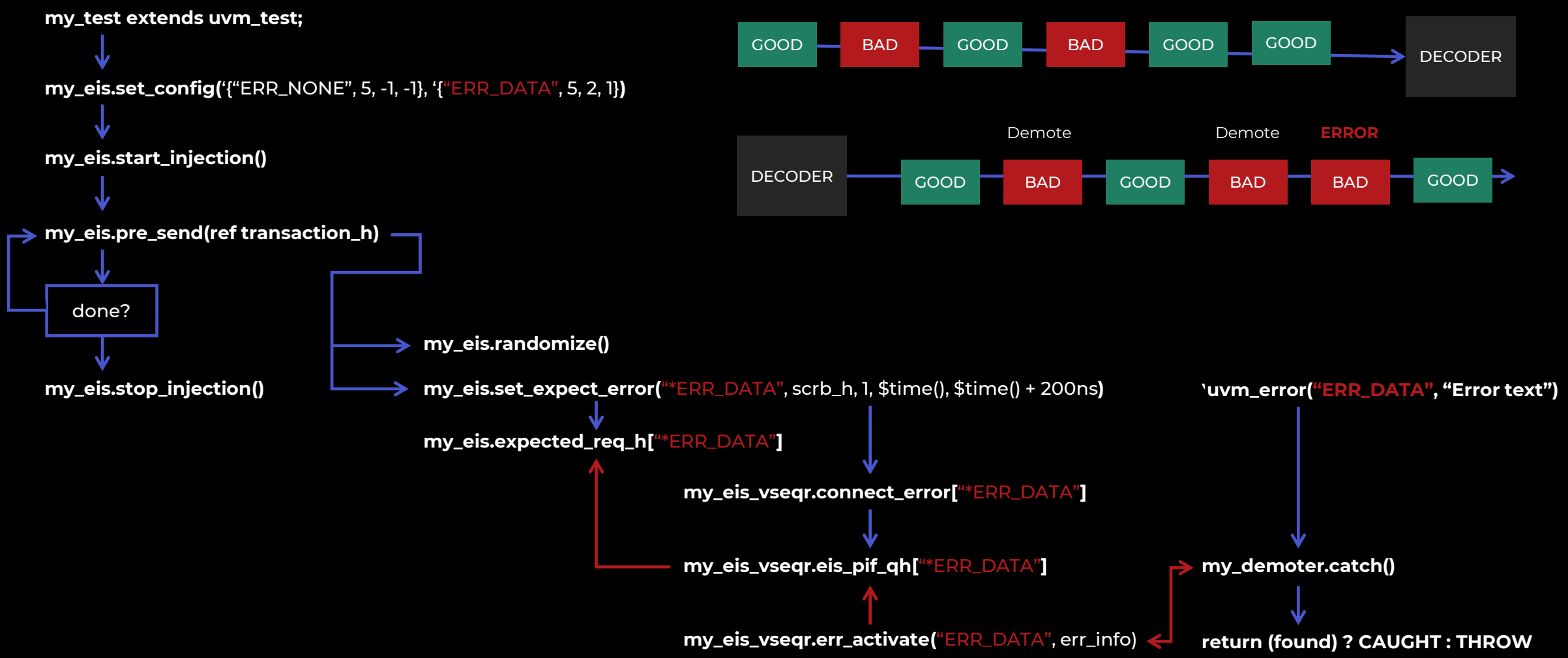

FPGA SYSTEMS

Пример использования «JESD»





Пример использования «DECODER»



Итог

Данный подход является универсальным и не требует серьезных модификаций исходного кода IP

Механизм добавления ошибок позволяет применять подход Constraint Random Verification

Понятный механизм добавления ошибок и контроля результата

БЮРО 1440

1440.space