



Настройки SVEEP

Чтобы включить SVEEP (SystemVerilog Ethernet Endpoint), нужно задать следующие настройки.

+cfg.has_eth_env=1	Включает UVM-среду Ethernet. По умолчанию активны все 6 портов. Если нужна иная конфигурация, см. #12153-10, уровень агента.
+sw_test+enable_sveep	Включает ответную часть SVEEP.
+SVEEP/AGENT +SVEEP/IF +SVEEP/RX/FAME +SVEEP/TX/FAME	Включает печать от соответствующего слоя SVEEP.
-sv_root <PATH> -sv_lib <LIB>	Настройки sv_root/sv_lib совместно определяют используемый so-файл с программным сценарием. Его нужно скомпилировать заранее. PATH указывает на путь (абсолютный или относительный от директории запуска) до so-файла, исключая сам файл, а LIB задаёт имя файла без расширений.

```
export GIT_HOME=/mnt/nas3-ssdvl/users/gabidullin/hornet3/;
export VIP_HOME=${GIT_HOME}/unitverif/top/soc_tb/vip;

export RTL_DIR=/mnt/nas3-bamboo-agents/BUILD_DIR_BAMBOO/rtl_builds/hornet3/HRN2_RTL_SOCUVM_soc_2021-08-03_18-46-31_7b819af_TB_u_2021-07-29_11-22-42_c5dbc1a_/RTL_UNIT_RTL_SMS_0-2018.09-SP2-3;

SIM_OPTS+=" +cfg.has_eth_env=1 +sw_test+enable_sveep -sv_root /mnt/nas3-ssdvl/users/gabidullin/tests/ethernet/sveep/script -sv_lib scenario +SVEEP/AGENT +SVEEP/IF +SVEEP/RX/FAME +SVEEP/TX/FAME ";

make -f $GIT_HOME/sysverif/run/scenario/makefile TEST=ethernet/receive CORE_NAME=CPU0 DEST=SRAM_ONE EXTEND_UVM_SIM_OPTS+=" $SIM_OPTS +timeout_us=999999999999 " █ PARAMS=" ETH_SCENARIO=ETH_SC_14 ETH_PACKETS=1
ETH_PACKET_LEN=128 LOG_ETH_TEST ETH_MODE=ETH_MODE_RGMII BM_LOG_DEBUG LOG_BM_HSR BM_LOG_TRACE LOG_BM_ETH_API " TEST_DIR=. CROSS_PATH=/mnt/nfs/store/x-tools/riscv32-unknown-elf-7.1.1/bin/ CONFIG=rtl re
buildBin sim
~
```

```
PROJECT_HOME=/mnt/nas3-ssdv1/users/gabidullin/tests/ethernet/sweep
```

```
gcc -Werror -fpic -I$PROJECT_HOME/c -c $PROJECT_HOME/c/util.c $PROJECT_HOME/c/scenario.c  
gcc -shared -o scenario.so util.o scenario.o
```

```
extern int sveep_transmit(int *status, int port_num, char *tx_buf, int length, int append_fcs);
extern int sveep_receive(int *status, char *rx_buf, int *length, int port_num, int timeout_us);
extern int sveep_sleep_us(int value);
void test_tx(int *status, int port_num);
void test_rx(int *status, int port_num);
int sveep_init(const int version);
int sveep_run(int *status, const int port_num);
```

```
int sveep_init(const int version)
{
    printf("[C] Entry to sveep_init(0x%x)\n", version);
    return 0;
}
```

DPI-часть вызывает эту функцию, чтобы инициализировать программный сценарий.

```
int sveep_run(int *status, const int port_num)
{
    int status_tx;
    int status_rx;

    printf("[C] Entry to sveep_run(%d)\n", port_num);
    sveep_sleep_us(1000 * (port_num + 1));
    test_tx(&status_tx, port_num);
    test_rx(&status_rx, port_num);

    *status = status_tx + status_rx;

    printf("[C] Exit from sveep_run(%d, %d)\n", *status, port_num);
    return 0;
}
```

1. Скомпилировать динамическую библиотеку .SO

а) объявить и определить методы: `sveep_init`, `sveep_run`, `test_tx`, `test_rx`

б) объявить прототипы методов: `sveep_transmit`, `sveep_receive`,
`sveep_sleep_us`

б) компиляция .SO

2. Строка запуска

а) указать в строке запуска путь к .SO библиотеки

б) выполнить строку запуска