Core DPI verification

Example of shared-memory usage:

use-shared-memory-1.c

use-shared-memory-2.c

Fast tutorial of shared-memory usage:

aos201617_multiprocess_programming_updated20161223.pdf

To use emusrup for verification, one needs to execute in SV testbech functions as follows:

1. tbEmusrupStart - initializes emusrup for DPI debugging.
2. tbEmusrupProceed - emusrup proceeds a single step of the code.
3. tbEmusrupCheckLogic - returns values of logical registers in srup.
4. tbEmusrupCheckShadow - returns values of shadow registers in srup.
5. tbEmusrupStop - finalizes emusrup execution for DPI debugging.

When tbEmusrupStart is called from DPI testbench, the arrays for storing handle and semaphore must be allocated at SV side. It stems from the fact that SV does not provide (**char) type allowing to return pointer to allocated memory. Only equivalent data types string(char *) are available. Hence, memory for string must be allocated in SV testbench.

Emusrup code is located in ./software/emusrup directory whereas its compilation for DPI verification is executed by script in ./software/DPI/emusrup_debug directory. Results of compilation go to ./firmware/sim/lib directory.

```
/** Function starts execution of emusrup in DPI debug mode.
 * \param dir directory with simulation files
 * \param lfsr switch for lfsr order useage
 * \param num_of_addr_bits size of lfsr shift register
 * \param handle name of shared memory file
 * \param semaphore name of semaphore file
 */
import "DPI-C" function void tbEmusrupStart (
 input string dir,
 input int lfsr,
 input int num_of_addr_bits,
 output string handle,
 output string semaphore
 );

/** Function executes a single step of emusrup in DPI debug mode.
 * \param handle name of shared memory file
 * \param semaphore name of semaphore file
 * \param opcode_instr code of executed instruction
 */
import "DPI-C" function void tbEmusrupProceed (
 input string handle,
 input string semaphore,
 output int opcode_instr
);

/** Function returns status of logical registers of emusrup in DPI debug
mode.
 * \param handle name of shared memory file
 * \param semaphore name of semaphore file
 * \param data_logic_addr logical number of checked register
 * \param data MPA number in half-integer form
 * \param data_prec precision of the number in register (in number of
limbs)
```

```
 * \param data_sign sign of the number
 * \param data_phys_addr physical number of checked register
 */
import "DPI-C" function void tbEmusrupCheckLogic (
 input string handle,
 input string semaphore,
 input int data_logic_addr,
 output int data[0:(4*MAX_PREC)-1],
 output int data_prec,
 output int data_sign,
 output int data_phys_addr
);

/** Function returns status of shadow registers of emusrup in DPI debug
mode.
 * \param handle name of shared memory file
 * \param semaphore name of semaphore file
 * \param data_shadow_addr logical number of checked register
 * \param data MPA number in half-integer form
 * \param data_prec precision of the number in register (in number of
limbs)
 * \param data_sign sign of the number
 * \param data_phys_addr physical number of checked register
 */
import "DPI-C" function void tbEmusrupCheckShadow (
 input string handle,
 input string semaphore,
 input int data_shadow_addr,
 output int data[0:(4*MAX_PREC)-1],
 output int data_prec,
 output int data_sign,
 output int data_phys_addr
);

/** Function stops execution of emusrup in DPI debug mode.
 * \param handle name of shared memory file
 * \param semaphore name of semaphore file
 */
import "DPI-C" function void tbEmusrupStop (
```

```
  input string handle,
  input string semaphore
);
```

Data storage in memory example

limb_size = 8 bits

4 limb data = 0 x 12 34 56 78

data_size = 3 (no lfsr) or 8 (lfsr)

data_size equals to the pointer of the last data portion

address	no LFSR	LFSR		data size [limbs]	no LFSR	LFSR
0	78	0		0 - impossible *	0	1
1	56	78		1	1	2
2	34	56		2	2	4
3	12	0		3	3	8
4	0	34		4	4	16
5	0	0		5	5	33
6	0	0		6	6	66
7	0	0		7	7	132
8	0	12		8	8	264
9	0	0		9	9	17
...	0	0		...	...	...

* - 0 limbs is represented as 1 limb data of value 0.