Configuration parameters for **FlexGrip**

Kernel configuration parameters:

The ***pickbench.vhd*** file defines the kernel parameters to be configured as fixed during the GPGPU execution. Those parameters are partially defined in the CUDA kernel program by the user and other by the configuration controller in the compilation process. However, the lack of an interpreter of the NVIDIA'S configuration controller requires the manual definition of those parameters.

```
3
4   library IEEE;
5   use IEEE.STD_LOGIC_1164.ALL;
6   use IEEE.numeric_std.all;
7   -- Uncomment the following library declaration if using
8   -- arithmetic functions with Signed or Unsigned values
9   --use IEEE.NUMERIC_STD.ALL;
10
11  package pick_bench is
12
13      type cmem_regs_type   is array (63 downto 0) of std_logic_vector(7 downto 0);
14      type kernel_regs_type is array (15 downto 0) of std_logic_vector(31 downto 0);
15
16
17      constant BENCH_CORES : integer := 32;
```

The **BENCH CORES** can be selected among the options:
32, **16** and **8**
It defines the total number of SP cores to be included in the compilation of **FlexGrip**.

```
18      constant BENCH_WARP_LANES : integer := 1;
```

The **BENCH_WARP_LINES** defines the ratio: WARP_SIZE/BENCH_CORES
WARP_SIZE is fixed by hardware constraints

Nevertheless, this parameter must be carefully selected considering that in some applications (Those including more than one block, the dispatcher may consider the total number of threads to be distributed in a SP as more as expected. A method to avoid this behavior is based on the restriction of the bench warp lines assigned to each SP core. (The range is between 1 and 4)

This ratio defines the total number of warp lines required depending on the cores selected:

BENCH CORES = 32 BENCH_WARP_LINES = 1
BENCH CORES = 16 BENCH_WARP_LINES = 2
BENCH CORES = 8 BENCH_WARP_LINES = 4

```
20      constant BENCH_APP : string := "TP";      -- vector_add (256 block - 8 warps)
21      constant BENCH_APP_INST : string := "TP";
```

The **BENCH_APP** parameter defines the Flexgrip application to be selected and configured. In previous Flexgrip versions just a limit number of applications were developed. For new app, the best option is TP (Test Program).

```
23      -- parametros del kernel, dependen de la aplicacion, address and limit count
24      constant BENCH_KREG0: std_logic_vector(31 downto 0) := x"00000000";
25      constant BENCH_KREG1: std_logic_vector(31 downto 0) := x"00000100";
26      constant BENCH_KREG2: std_logic_vector(31 downto 0) := x"00000200";
27      constant BENCH_KREG3: std_logic_vector(31 downto 0) := x"00000008";
```

The **BENCH_KREG0**, **BENCH_KREG1**, **BENCH_KREG2** and **BENCH_KREG3** define the kernel input parameters.

A traditional Cuda kernel description employs the syntaxes:
<<<***Threads_per_block***, ***Blocks_per_Grip***>>>kernel_name(***input1**, ***input2**, ***input3**)

BENCH_KREG0 belong to the memory location of **input1**, specified in Hex notation.
BENCH_KREG1 belong to the memory location of **input2**, specified in Hex notation.
BENCH_KREG2 belong to the memory location of **input3**, specified in Hex notation.
BENCH_KREG3 can be used as an integer constant for the kernel.

```
constant BENCH_K_GRDX : std_logic_vector(15 downto 0) := x"0002";
constant BENCH_K_GRDY : std_logic_vector(15 downto 0) := x"0001";
```

A traditional Cuda kernel description employs the syntaxes:
<<<***Threads_per_block***, ***Blocks_per_Grip(x,y)***>>>kernel_name(***input1**, ***input2**, ***input3**)

The parameters **BENCH_K_GRDX** and **BENCH_K_GRDY** defines the grid dimension of the program kernel. As the total number of blocks to be executed by the model.
In the syntax those parameters are replaced by ***Blocks_per_Grip(x,y)***

```
constant BENCH_K_BLKX : std_logic_vector(15 downto 0) := x"0020";
constant BENCH_K_BLKY : std_logic_vector(15 downto 0) := x"0001";
constant BENCH_K_BLKZ : std_logic_vector(15 downto 0) := x"0001";
```

A traditional Cuda kernel description employs the syntaxes:
<<<***Threads_per_block(x,y,z)***, ***Blocks_per_Grip(x,y)***>>>kernel_name(***input1**, ***input2**, ***input3**)

The parameters **BENCH_K_BLKX**, **BENCH_K_BLKY** and **BENCH_K_BLKZ** define and represent the total number of threads per block to be executed on FlexGrip.
In the syntax those parameters are replaced by ***Threads_per_block(x,y,z)***

```
constant BENCH_CMEM_PARAM_SIZE          : std_logic_vector(5 downto 0)  :=  "100000";
constant BENCH_KERNEL_PARAM_SIZE        : std_logic_vector(3 downto 0)  :=  x"4";
constant BENCH_KERNEL_GPRS              : std_logic_vector(8 downto 0)  :=  "000100000";
constant BENCH_KERNEL_SHMEM_SIZE        : std_logic_vector(31 downto 0) :=  x"0000002c";
constant BENCH_BLOCKS_PER_CORE          : std_logic_vector(3 downto 0)  :=  x"1";
```

The **BENCH_CMEM_PARAM_SIZE** defines the size of the constant memory for FlexGrip.
The **BENCH_KERNEL_PARAM_SIZE** defines the total number of input kernel parameters in the program (***BENCH_KREG0***, ***BENCH_KREG1***, ***BENCH_KREG2*** and ***BENCH_KREG3***)

The **BENCH_KERNEL_GPRS** defines the size of the general purpose register on each register file associated to each thread. (*registers-per-thread*). Depending on the application this value should be changed and computed.

The **BENCH_KERNEL_SHMEM_SIZE** defines the size of the shared memory lines associated to each thread. (*shared_memory-per-thread*). Depending on the application this value should be changed and computed.

The **BENCH_BLOCKS_PER_CORE** defines the total number of blocks that can be assigned and executed (simultaneously) in the SM of Flexgrip.

This parameter by default must be 8, but depending on the application, this value may change the results. Originally, the **Block ID** parameter was designed to be read from the Shared Memory. But, it seems that the actual version of the memory hierarchy in FlexGrip is not available to change this constant values. Thus, this issue was partially solved by changing this value in the block scheduler. (Checking this condition)

```
    constant cmem_regs_default : cmem_regs_type := ( -- 63(3F)
        x"00", x"00", x"00", x"00",                  -- 60
        x"00", x"00", x"00", x"00",                  -- 56
        x"00", x"00", x"00", x"00",                  -- 52
        x"00", x"00", x"00", x"00",                  -- 48
        x"00", x"00", x"00", x"00",                  -- 44
        x"00", x"00", x"00", x"00",                  -- 40
        x"00", x"00", x"00", x"00",                  -- 36
        x"00", x"00", x"00", x"00",                  -- 32
        x"00", x"00", x"00", x"00",                  -- 28
        x"00", x"00", x"00", x"00",                  -- 24
        x"00", x"00", x"00", x"00",                  -- 20
        x"00", x"00", x"00", x"00",                  -- 16
        x"00", x"00", x"00", x"00",                  -- 12
        x"00", x"00", x"00", x"00",                  -- 8
        x"00", x"00", x"00", x"00",                  -- 4
        x"00", x"00", x"03", x"ff");                 -- 0

    -- ORIGINAL FOR TP:
    -- x"03", x"ff", x"03", x"ff"

    constant kernel_regs_default : kernel_regs_type := (     -- 15 (0F)
        x"00000000", x"00000000", x"00000000", x"00000000",  -- 12
        x"00000000", x"00000000", x"00000000", x"00000000",  -- 8
        x"00000000", x"00000000", x"00000000", x"00000000",  -- 4
        x"00000000", x"00000040", x"00000200", x"00000000"); -- 0
```