Demonstration of RAVAC compilation for the ASVP/EdkDSP Platform

Virtual machine installation and login

Virtual machine was created with Virtual Box v. 4.2.6, also any newer version will work. Through File->Import Appliance, the virtual machine image is installed.

After booting, the user name is `smecy`, password is also `smecy`. Root password if needed is `smecy` too.

Acceleration with RAVAC

The source codes for the FG/BG demonstrator are located in `~/edkdsp/_smecy_demonstrator_fgbg_mog/fgbg_mog_algorithm_dab`

The RAVAC compiler processes SMECY IR 1 pragmas; they are used to be able to use multiple ASVP accelerators at once. Example can be found in `fgbg_mog.c` in function `process_frame`. Before RAVAC, there was no compiler able to handle OpenMP pragmas for MicroBlaze.

Next extension are `#pragma smecy vectorize` pragmas. These specify code that will be offloaded to an ASVP accelerator. They can be found in file `accel_main.c`, where they specify fot loops that will be offloaded.

Running RAVAC

The RAVAC compiler is prepared in the `/home/smecy/edkdsp` directory. Demonstrator application can be found in `~/edkdsp/_smecy_demonstrator_fgbg_mog/fgbg_mog_algorithm_dab`.

There are two ways how to run compilation: with or without acceleration on BCEs.

`./build.sh` for accelerated compilation, and

`./build.sh –u` for unaccelerated compilation.

In both cases a binary `fgbg_mog` is generated. This binary can be the copied to the ASVP/EdkDSP platform and run.

The demonstration is prepared not to use the optional floating-point support in the Microblaze core and just to use the ASVP accelerators. If you would need to compile with floating-point support on Microblaze, uncomment line 42 in `~/edkdsp/ravac/ravac.sh` and line 4 in `~/edkdsp/_smecy_demonstrator_fgbg_mog/fgbg_mog_algorithm_dab/accel_main.sh`.

Running demonstrator on the ASVP platform

Once the binary is compiled using the RAVAC compiler, it must be together with input images copied to a tftp server directory. The whole `~/edkdsp/_smecy_demonstrator_fgbg_mog/ and ~/edkdsp/fpga_coonfig` must be copied to a host computer that is connected to the ASVP board and contains tools for FPGA programming (WebISE, specifically impact), and picocom terminal. Such environment should be provided by the platform provider and this should be preferably done by someone that knows how to program the ASVP board.

This could not have been prepared on the VM since it needs to access hardware peripherals.

Following commands prepare data and binary in the tftp server.

```
cd ~/edkdsp/_smecy_demonstrator_fgbg_mog/fgbg_mog_algorithm_dab/

cp inputs/scene01_small/* /tftp/

cp fgbg_mog /tftp/
```

Now the board is configured:

```
cd ~/edkdsp/fpga_config

./dev_conf.sh
```

Then connect to the device, permission change may be needed:

chmod o+rw /dev/ttyUSB0

picocom -b 115200 /dev/ttyUSB0

And load and boot petaLinux (tftp server must be running on your PC, if you are getting Permission denied error messages, disable SELinux)

UBoot> tftp 0x61000000 image.ub

bootm

Login to the running petalinux with

root:root

The /tmp directory has read/write permissions, you can now load your needed binary and data. The commands span over multiple lines here.

```
cd tmp
```

```
echo "get scene01_f113.png" | tftp 192.168.0.1 ; echo "get
scene01_f114.png" | tftp 192.168.0.1 ; echo "get scene01_f115.png" |
tftp 192.168.0.1 ; echo "get scene01_f116.png" | tftp 192.168.0.1 ;
echo "get scene01_f117.png" | tftp 192.168.0.1
```

```
echo "get scene01.list_local" | tftp 192.168.0.1 ; echo "get fgbg_mog"
| tftp 192.168.0.1 ; chmod 755 fgbg_mog; mkdir out
```

To take advantage of multiple accelerators, let's use multiple threads:

```
export OMP_NUM_THREADS=2
```

Now the compiled application is run:

```
./fgbg_mog scene01.list_local out/
```

Resulting images are stored into the out directory, they can be sent back to the host PC by running:

```
cd out; echo "put mask_000.png" | tftp 192.168.0.1 ;  echo "put
mask_001.png" | tftp 192.168.0.1 ; echo "put mask_002.png" | tftp
192.168.0.1 ; echo "put mask_003.png" | tftp 192.168.0.1 ; echo "put
mask_004.png" | tftp 192.168.0.1; cd ..
```

It is possible, that the tftp server will print permission denied message, in this case, on the host PC run

```
touch /tftp/mask_000.png; touch /tftp/mask_001.png; touch
/tftp/mask_002.png; touch /tftp/mask_003.png; touch /tftp/mask_004.png;
chmod 777 /tftp/mask*
```

And repeat the output images transfer.