IBM CAPI SNAP framework

Version 1.0

How to Create a New Action in SNAP environment.

The Guide describes how to create a new action in SNAP environment.

Overview

Let's imagine that you want to create a new action for which files used will be as follow:

- Application ➔ snap_newaction.c Directory: actions/hls_newaction/**sw**
- Software action ➔ newaction_software.c Directory: actions/hls_newaction/**sw**
- Hardware action ➔ newaction_hardware.cpp Directory: actions/hls_newaction/**hw**
- Common header file ➔ newaction_commonheader.h Directory: actions/hls_newaction/**include**

The simplest is to start from an existing example that contains the **same interfaces to external resources** you need. This will setup all the access you will need for your algorithm.

We will use the Nimbix cloud environment to illustrate the changes but this can be easily translated to any other environment.

This document will succesfully go through the following items:
- install and setup the SNAP environment
- create a ***newaction*** action by duplicating and adapting an existing action
- run a modelization of the new action

Related documentation

Quick Start Guide on a General environment

How to debug an issue in SNAP environment

How to optimize a function in SNAP environment

Can be found in **https://github.com/open-power/snap/doc**

Contents

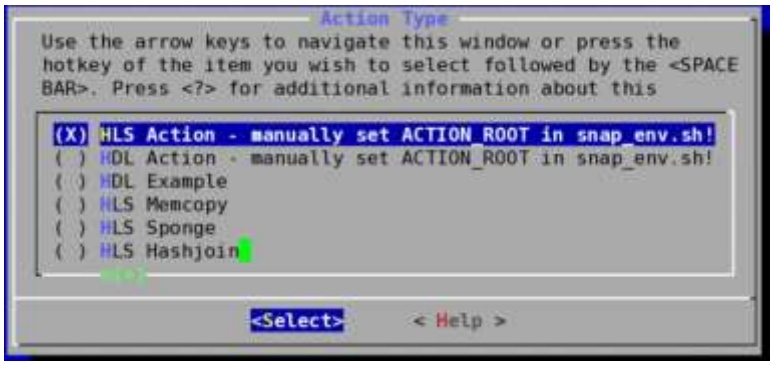
1. Environment setup

Let's first install the SNAP framework that can be downloaded from github:

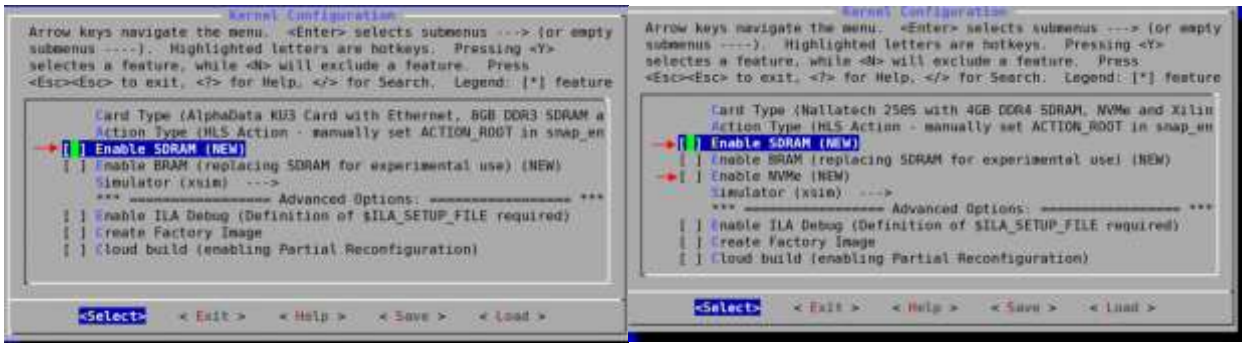
```
cd
git clone https://github.com/open-power/snap
cd ~/snap
cp ~/snap.env.sh .          (<= Nimbix specific)
make snap_config
```

Choose the card that will contain the resources you need

then select HLS Action type.



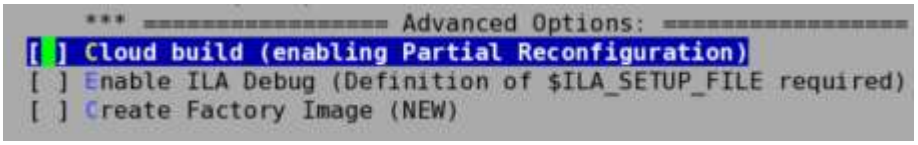
Select then the different resources of the card you have selected and that you need to use for your algorithm (ADKU3 on left – N250S on right)



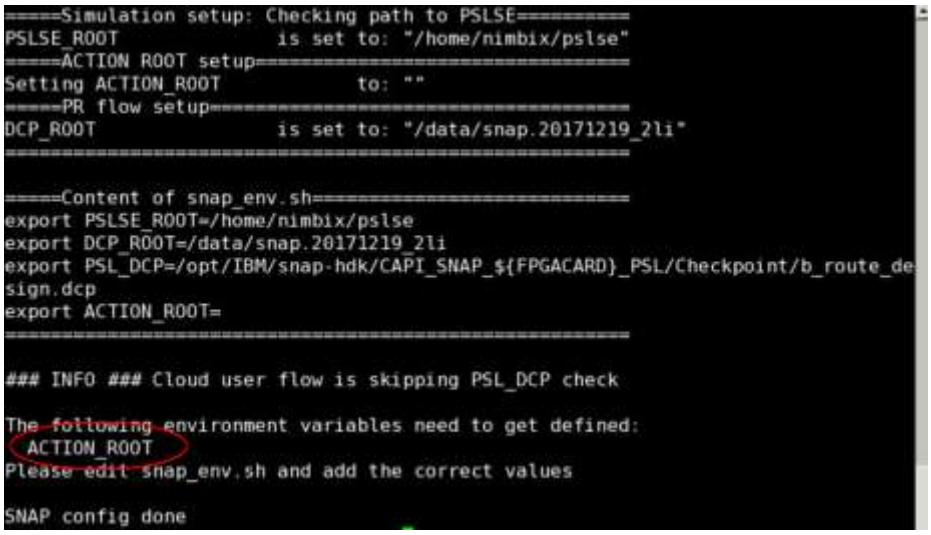
VERY IMPORTANT:
Be coherent with the resources you are selecting. If you select the SDRAM or NVMe, you will need to ensure that they are enabled in the hardware action ports (function hls_action of the hardware action).

It makes sense as SNAP will attach the physical hardware drivers you selected to your action.

On Nimbix, don't forget to select the Cloud Build option



After exiting the snap_config, you should get the following screen if this is the first time you use the snap_config, otherwise you may have ACTION-ROOT set to the previous configuration



Edit **~/snap/snap_env.sh** and add the path to the new action directory

From snap directory type :

`Make software` to make sure we have all the libs prepared for next "make" steps

2. Choose an NEWACTION_TYPE id so that your action will be identified

Edit ***~snap/ActionTypes.md*** and pick up a new number in the file. Let's get for example 00.00.00.01. You will be able to keep it for your internal use or follow the process explained at the bottom of this document to get a unique number.

```
## Action Type Assignment
Vendor | Range Start | Range End | Description
:--- | :--- | :--- | :---
Reserved | 00.00.00.00 | 00.00.00.00 | Reserved
free | 00.00.00.01 | 00.00.00.01 | HLS My New Action
free | 00.00.00.02 | 00.00.FF.FF | Free for experimental use
IBM | 10.14.00.00 | 10.14.00.00 | SNAP framework example
IBM | 10.14.00.01 | 10.14.00.01 | HDL NVMe example
```

3. Copy and adapt an existing action example to a new action

3.1 Copy an existing hls example

Let's take the hls_helloworld as the example from which we will start from. You can take any of the examples or even create a new one copying the different Makefile and the stucture of directories.

```
cd actions
cp -r hls_helloworld hls_newaction
cd hls_newaction
rm doc/*                          # <= cleaning unrelated stuff
rm tests/*                        # <= cleaning unrelated stuff
```

3.2 Adapt the example : include directory

```
(cd ~/snap/actions/hls_newaction)
cd include
```

Rename the copied header file to ***newaction_commonheader.h.*** You should so have

```
[nimbix@JARVICENAE-0A0A1856 include]$ mv action_changecase.h
newaction_commonheader.h
 [nimbix@JARVICENAE-0A0A1856 include]$ ls
newaction_commonheader.h
```

Edit this file and you will have to adapt the following things:

- ACTION_TYPE value ➔ change the name to **NEWACTION_ACTION_TYPE** and set its value to the one you choosed in the ActionTypes.md file. Let's say you took for example 00.00.00.01

```
/* This number is unique and is declared in ~snap/ActionTypes.md */
#define NEWACTION_ACTION_TYPE 0x00000001
```

- You will also have to change the structure name to **newaction_job** and its **content** to the data you have decided to exchange between the application and the action.

```
typedef struct newaction_job {
        struct snap_addr in;    /* input data */
        struct snap_addr out;   /* offset table */
} newaction_job_t;
```

Don’t also forget to update the following:

- change the #ifndef/#define at the beginning of the file ***newaction_commonheader.h***.
- **later**, don’t forget to change also all reference to helloworld variables and algorithm

NiceToHave:

➔ If you want to have your action appearing in the kconfig menu, see **Annex 1** of this document.

3.3 Adapt the example : sw directory

```
cd ../sw
```

Rename the copied action file to ***newaction_software.c*** and the application file to ***snap_newaction.c.*** You should so get :

```
[nimbix@JARVICENAE-0A0A1856 sw]$ mv action_lowercase.c
newaction_software.c
[nimbix@JARVICENAE-0A0A1856 sw]$ mv snap_helloworld.c
snap_newaction.c
[nimbix@JARVICENAE-0A0A1856 sw]$ ls
Makefile  newaction_software.c  README.md  snap_newaction.c
```

Edit ***Makefile*** and update the name of the software action file(s) (newaction_software.o) and the name of the application (snap_newaction)

```
# This is solution specific. Check if we can replace this by generics too.

snap_newaction: newaction_software.o          # software action file
snap_newaction_objs = newaction_software.o    # software action file

projs += snap_newaction                       # application file
```

Don't forget to update the following:

- in ***newaction_software.c*** file
  - change the include to ***newaction_commonheader.h*** and
  - change all references (1 occurrence) to HELLOWORLD_ACTION_TYPE variable to **NEWACTION_ACTION_TYPE**
  - change all references (2 occurrences) to helloworld_job by **newaction_job**
  - **later**, don't forget to change also all reference to helloworld variables and algortithm

```
[nimbix@JARVICENAE-0A0A1860 sw]$ diff newaction_software.c
../../hls_helloworld/sw/action_lowercase.c
36c36
< #include <newaction_commonheader.h>
---
> #include <action_changecase.h>
58c58
<       struct newaction_job *js = (struct newaction_job *)job;
---
>       struct helloworld_job *js = (struct helloworld_job *)job;
97c97
<       .action_type = NEWACTION_ACTION_TYPE, // Adapt with your ACTION NAME
---
>       .action_type = HELLOWORLD_ACTION_TYPE, // Adapt with your ACTION NAME
```

- in ***snap_newaction.c*** file
  - change the include to ***newaction_commonheader.h*** and
  - change all references to helloworld_job by **newaction_job**
  - **later**, don't forget to change also all reference to helloworld variables and algortithm

```
[nimbix@JARVICENAE-0A0A1860 sw]$ diff snap_newaction.c
../../hls_helloworld/sw/snap_helloworld.c
38c38
< #include <newaction_commonheader.h>
---
> #include <action_changecase.h>
82,83c82,83
<       "SNAP_CONFIG=FPGA $ACTION_ROOT/sw/snap_newaction -i /tmp/t1 -o /tmp/t2\n"
<       "SNAP_CONFIG=CPU  $ACTION_ROOT/sw/snap_newaction -i /tmp/t1 -o /tmp/t3\n"
---
>       "SNAP_CONFIG=FPGA $ACTION_ROOT/sw/snap_helloworld -i /tmp/t1 -o /tmp/t2\n"
>       "SNAP_CONFIG=CPU  $ACTION_ROOT/sw/snap_helloworld -i /tmp/t1 -o /tmp/t3\n"
98c98
<                               struct newaction_job *mjob,
---
>                               struct helloworld_job *mjob,
134c134
<       struct newaction_job mjob;
---
>       struct helloworld_job mjob;
327c327
<       action = snap_attach_action(card, NEWACTION_ACTION_TYPE, action_irq, 60);
---
>       action = snap_attach_action(card, HELLOWORLD_ACTION_TYPE, action_irq, 60);
```

Check that everything is ok by typing

```
make
```

if you have "….pslse/libcxl/libcxl.h:21:22: error: misc/cxl.h: No such file or directory" error, you can run a make software from the snap directory, this will compile the libcxl library. Then come back in /sw directory and run make again.

You should get:

```
[nimbix@JARVICENAE-0A0A1860 sw]$ ls
Makefile              newaction_software.o  snap_newaction
newaction_software.c  README.md             snap_newaction.c
```

Then execute the action

```
./snap_newaction
```

```
[nimbix@JARVICENAE-0A0A1844 sw]$ ./snap_newaction
Usage: ./snap_newaction [-h] [-v, --verbose] [-V, --version]
  -C, --card <cardno>       can be (0...3)
  -i, --input <file.bin>    input file.
  -o, --output <file.bin>   output file.
  -A, --type-in <CARD_DRAM, HOST_DRAM, ...>.
  -a, --addr-in <addr>      address e.g. in CARD_RAM.
  -D, --type-out <CARD_DRAM,HOST_DRAM, ...>.
  -d, --addr-out <addr>     address e.g. in CARD_RAM.
  -s, --size <size>         size of data.
  -t, --timeout             timeout in sec to wait for done.
  -X, --verify              verify result if possible
  -N, --no-irq              disable Interrupts
```

Simple example to test it is for example

```
echo “hello World. This is my first CAPI SNAP experience.” > /tmp/t1
SNAP_CONFIG=CPU ./snap_newaction -i /tmp/t1 -o /tmp/t2
cat /tmp/t1
hello World. This is my first CAPI SNAP experience.”
cat /tmp/t2
hello world. this is my first capi snap experience.”
```

3.4 Adapt the example : hw directory

```
cd ../hw
```

Change the name of your files to the new names. You should then get

```
[nimbix@JARVICENAE-0A0A1860 hw]$ ls
Makefile  newaction_hardware.cpp  newaction_hardware.H  README.md
```

Edit **Makefile** and change the name of the hardware action file (**newaction_hardware.cpp**), the name of the directory where Vivado HLS will generate your vhdl code (**hlsNewAction**) and the name of your action as the solution_name (**newaction**). These 2 last names can be set to anything since they are just internal names that user doesn't need to take care of.

```
# This is solution specific. Check if we can replace this by generics too.
SOLUTION_NAME ?= newaction     # internal name can be everything
SOLUTION_DIR ?= hlsnewaction   # internal name can be everything
srcs += newaction_hardware.cpp # hardware action file
```

```
[nimbix@JARVICENAE-0A0A1860 hw]$ diff Makefile ../../hls_helloworld/hw/Makefile
40,42c40,42
< SOLUTION_NAME ?= newaction     # internal name can be everything
< SOLUTION_DIR ?= hlsnewaction   # internal name can be everything
< srcs += newaction_hardware.cpp # hardware action file
---
> SOLUTION_NAME ?= helloworld
> SOLUTION_DIR ?= hlsUpperCase
> srcs += action_uppercase.cpp
```

Don't forget to update the followings:

- in ***action_hardware.cpp*** file
  - change the include to ***action_hardware.H*** and
  - change all references to HELLOWORLD_ACTION_TYPE variable to **NEWACTION_ACTION_TYPE**
  - **later**, don't forget to change also all reference to helloworld variables and algortithm

```
[nimbix@JARVICENAE-0A0A1860 hw]$ diff newaction_hardware.cpp
../../hls_helloworld/hw/action_uppercase.cpp
27c27
< #include "newaction_hardware.H"
---
> #include "action_uppercase.H"
111c111
<       Action_Config->action_type = NEWACTION_ACTION_TYPE; //TO BE ADAPTED
---
>       Action_Config->action_type = HELLOWORLD_ACTION_TYPE; //TO BE ADAPTED
```

- in ***action_hardware.H*** file
  - change the include to ***newaction_commonheader.h*** and
  - change all references to helloworld_job_t by **newaction_job_t**
  - update the #ifndef/#define at the beginning of the file.
  - **later**, don't forget to change also all reference to helloworld variables and algortithm

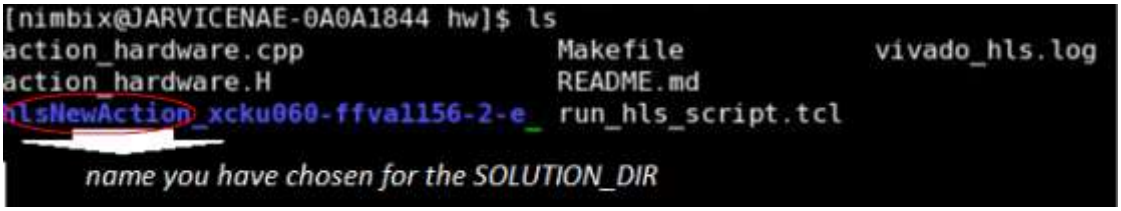
```
 [nimbix@JARVICENAE-0A0A1860 hw]$ diff newaction_hardware.H
../../hls_helloworld/hw/action_uppercase.H
25c25
< #include <newaction_commonheader.h> /* HelloWorld Job definition */
---
```

```
> #include <action_changecase.h> /* HelloWorld Job definition */
38,39c38,39
<       newaction_job_t Data;   /* up to 108 bytes */
<       uint8_t padding[SNAP_HLS_JOBSIZE - sizeof(newaction_job_t)];
---
>       helloworld_job_t Data;  /* up to 108 bytes */
>       uint8_t padding[SNAP_HLS_JOBSIZE - sizeof(helloworld_job_t)];
```

Check that everything is ok by typing:

```
make
```

You should get :



4. Build a simulation model and execute it

Once the changes done in the different files, let's see if everything is ok by building a simulation model:

```
cd ~/snap
make model
cd hardware/sim
./run_sim
```

Once the simulation window is opened you should be able to execute the discovery mode by typing **snap_maint -v.** If you have followed corretlty the previous changes then your action should be identified as you defined it in ActionTypes.md file

Calling then **snap_newaction** will show you the hls_helloworld information as we can expect it since we didn't change anything from the copied file

```
[nimbix@JARVICENAE-0A0A1844 20171219_151834]$ snap_newaction -h
Usage: snap_newaction [-h] [-v, --verbose] [-V, --version]
  -C, --card <cardno>       can be (0...3)
  -i, --input <file.bin>    input file.
  -o, --output <file.bin>   output file.
  -A, --type-in <CARD_DRAM, HOST_DRAM, ...>.
  -a, --addr-in <addr>      address e.g. in CARD_RAM.
  -D, --type-out <CARD_DRAM,HOST_DRAM, ...>.
  -d, --addr-out <addr>     address e.g. in CARD_RAM.
  -s, --size <size>         size of data.
  -t, --timeout             timeout in sec to wait for done.
  -X, --verify              verify result if possible
  -N, --no-irq              disable Interrupts
```

5. Deploy on P8 Machine

Deployment will be run as in the general case.

However unless your example is pushed into “snap” github repository, you’ll have to transfer the files on the new machine using /data common directory if you use Nimbix, or by any mean of your choice.

For example “snap_maint” snap tool won’t associate your new NEWACTION_TYPE number if it is not available in the snap repository.

ANNEX 1 : Add a New Action in the Kconfig menu

***NOTE:** we keep the fact that we copied the hls_helloworld action and will so keep the same resources.*

1. Edit **~/snap/scripts/Kconfig** to add the resources used which will be displayed in the menu

```
config HLS_HELLOWORLD
    bool "HLS HelloWorld"
    select ENABLE_HLS_SUPPORT
    select DISABLE_SDRAM_AND_BRAM
    select DISABLE_NVME

config HLS_NEWACTION
    bool "HLS NewAction"
    select ENABLE_HLS_SUPPORT
    select DISABLE_SDRAM_AND_BRAM
    select DISABLE_NVME
```

2. Edit **~/snap/snap_env** to add the path to the newaction

```
elif [ -n "$HLS_HELLOWORLD" ]; then
  AR='${SNAP_ROOT}/actions/hls_helloworld'
elif [ -n "$HLS_NEWACTION" ]; then
  AR='${SNAP_ROOT}/actions/hls_newaction'
```

You will notice that no SDRAM nor NVMe resources are displayed here since we disabled them in the kconfig file