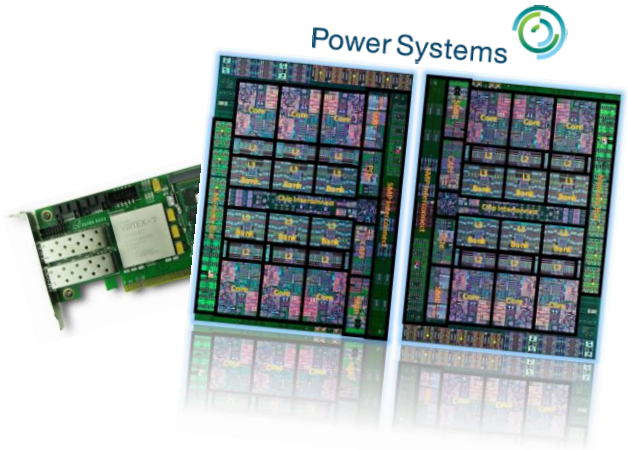
CAPI SNAP Education Series:
Module #3

CAPI SNAP Overview

"Let's get our hands dirty"

SNAP flow from A to Z
Step 2/3



September 20th , 2017

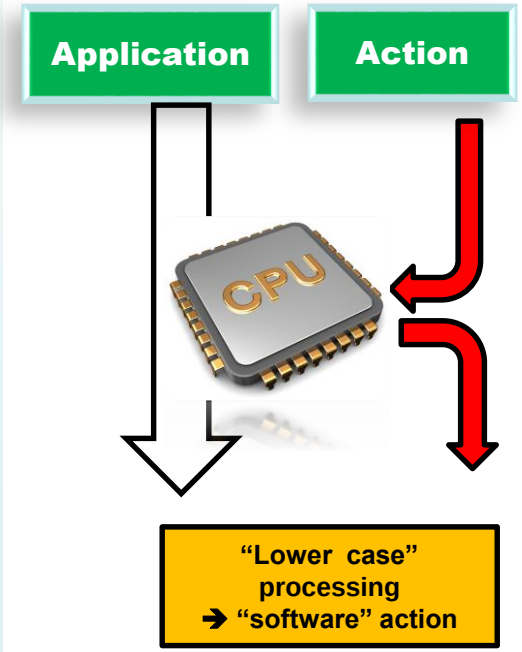
SNAP Framework built on Power™ CAPI technology

SNAP 3 steps flow

IBM Power Systems

1

SNAP_CONFIG=CPU
snap_**helloworld** –i /tmp/t1 -o /tmp/t2

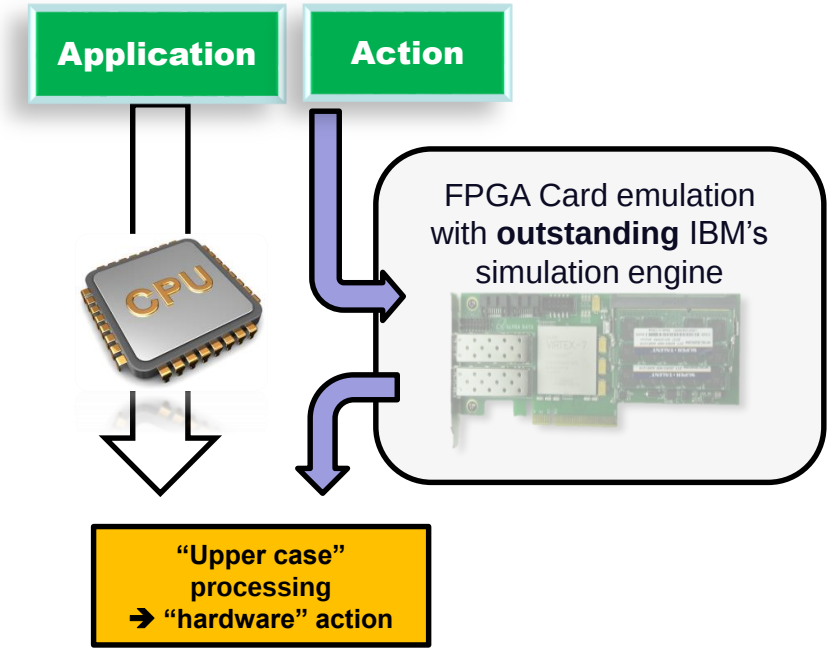


x86 server

➔*make*

2

SNAP_CONFIG=FPGA
snap_**helloworld** –i /tmp/t1 –o /tmp/t2

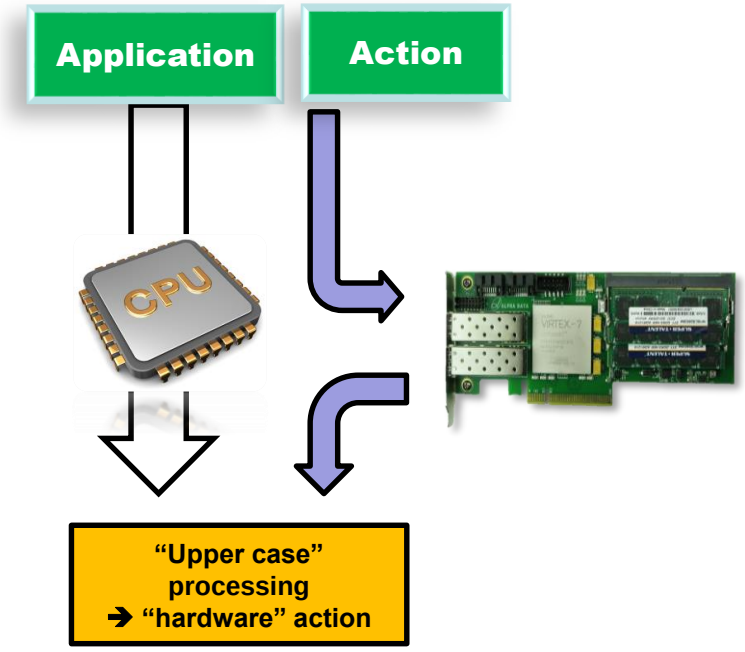


x86 server

➔*make sim*

3

SNAP_CONFIG=FPGA
snap_**helloworld** –i /tmp/t1 –o /tmp/t2



Power8 server

➔*make image*

Execution Phase
2nd STEP:
Create the "hardware" action code from the "software" action

1) **Identify and understand** what to modify
2) **Discover** the modified code
3) **Compile** and **execute** the code on CPU

main files:

- Application : snap_helloworld.c
  - software action : action_lowercase.c
  - hardware action : action_uppercase.cpp
  - *common header : action_changecase.h*

SNAP files organization:

SNAP
+ actions
+ hls_helloworld
+ sw *(cpu)*
- snap_helloworld.c
- action_lowercase.c
+hw *(fpga)*
- action_uppercase.cpp
+ include
- action_changecase.h

IDENTIFY
MODIFIED CODE
EXECUTION

Adapt action code for CPU to FPGA: new file

IBM Power Systems

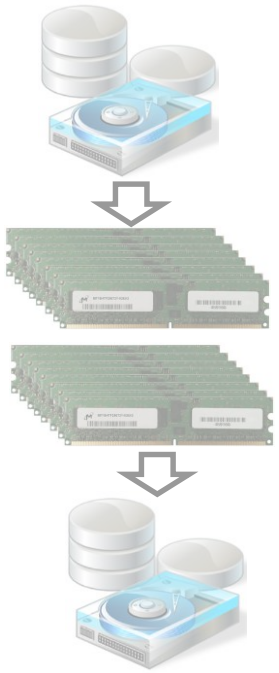
2

IDENTIFY

MODIFIED CODE

EXECUTION

snap_**helloworld** –i /tmp/t1 -o /tmp/t2 **-mode=fpga**



Start

Get input arguments to know files names

Allocate card
Attach action

Read data from ***input_file*** to **TextSource[ ]**

snap_prepare helloworld:
Addr_set(&Text_Source)
Addr_set(&Text_Dest)
Text_Length

Call function

Write data from ***DestText[ ]*** to ***output_file***

Print results

Detach action
Detach card

Exit

Working directory : hls_helloworld/sw

snap_helloworld.c

Read Input
&Text_Source
Text_Length

Upper case processing

Write Output
&Text_Dest
Text_Length

action_uppercase.cpp

1) Hardware action is created in a specific hw directory

SNAP files organization:

SNAP
+ actions
+ hls_helloworld
+ sw *(cpu)*
- snap_helloworld.c
- action_lowercase.c
+hw *(fpga)*
- action_uppercase.cpp
+ include
- action_changecase.h

Working directory : hls_helloworld/hw

Adapt action code for CPU to FPGA: getting data to process

IBM Power Systems

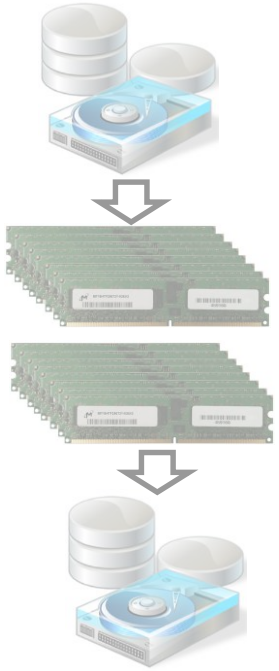
2

IDENTIFY

MODIFIED CODE

EXECUTION

snap_**helloworld** –i /tmp/t1 -o /tmp/t2 **-mode=fpga**



Start

Get input arguments to know files names

Allocate card
Attach action

Read data from ***input_file*** to **TextSource[ ]**

snap_prepare helloworld:
Addr_set(&Text_Source)
Addr_set(&Text_Dest)
Text_Length

Call function

Write data from ***DestText[ ]*** to ***output_file***

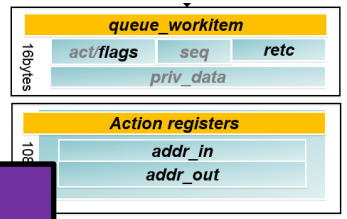
Print results

Detach action
Detach card

Exit

Working directory : hls_helloworld/sw

snap_helloworld.c



Get settings
Input_Address,
Output_Address and
number of bytes to process

Read Input
&Text_Source
Text_Length

Upper case processing

Write Output
&Text_Dest
Text_Length

1) Hardware action is created in a specific hw directory

2) Get data to process from specific structure

🡐 Action requests data from Source
SNAP framework does the access

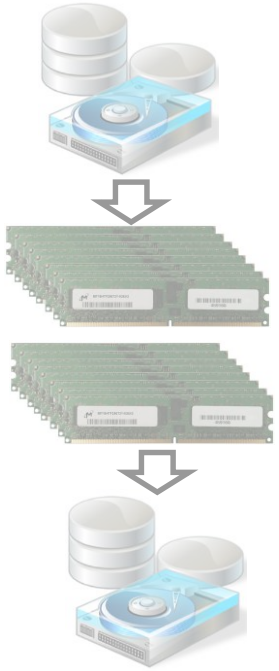
🡐 Action requests store to Destination
SNAP framework does the store

action_uppercase.cpp

Working directory : hls_helloworld/hw

Adapt action code for CPU to FPGA: add hls_action function

snap_**helloworld** –i /tmp/t1 -o /tmp/t2 **-mode=fpga**



Start

Get input arguments to know files names

Allocate card
Attach action

Read data from ***input_file*** to **TextSource[ ]**

snap_prepare helloworld:
Addr_set(&Text_Source)
Addr_set(&Text_Dest)
Text_Length

Call function

Write data from ***DestText[ ]*** to ***output_file***

Print results

Detach action
Detach card

Exit

Working directory : hls_helloworld/sw

snap_helloworld.c

hls_action

Get settings
Input_Address,
Output_Address and
number of bytes to process

Read Input
&Text_Source
Text_Length

Upper case processing

Write Output
&Text_Dest
Text_Length

action_uppercase.cpp

Working directory : hls_helloworld/hw

1) Hardware action is created in a specific hw directory

2) Get settings from specific structure

3) Add hls_action function

2 main functions:
- *hls_action*
- *process_action*

IDENTIFY

MODIFIED CODE

EXECUTION

Adapt action code for CPU to FPGA: looking to the code

IBM Power Systems

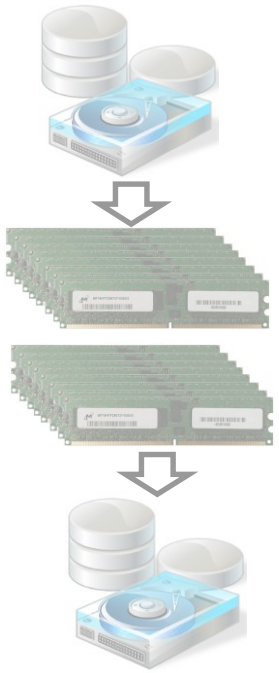
2

IDENTIFY

MODIFIED CODE

EXECUTION

snap_**helloworld** –i /tmp/t1 -o /tmp/t2 **-mode=fpga**



Start

Get input arguments to know files names

Allocate card
Attach action

Read data from ***input_file*** to **TextSource[ ]**

snap_prepare helloworld:
Addr_set(&Text_Source)
Addr_set(&Text_Dest)
Text_Length

Call function

Write data from ***DestText[ ]*** to ***output_file***

Print results

Detach action
Detach card

Exit

Working directory : hls_helloworld/sw

snap_helloworld.c

hls_action

Get settings
Input_Address,
Output_Address and
number of bytes to process

Read Input
&Text_Source
Text_Length

Upper case processing

Write Output
&Text_Dest
Text_Length

action_uppercase.cpp

hls_action function

process_action function

Let's identify all that in the code

2 main functions:
- *hls_action*
- *process_action*

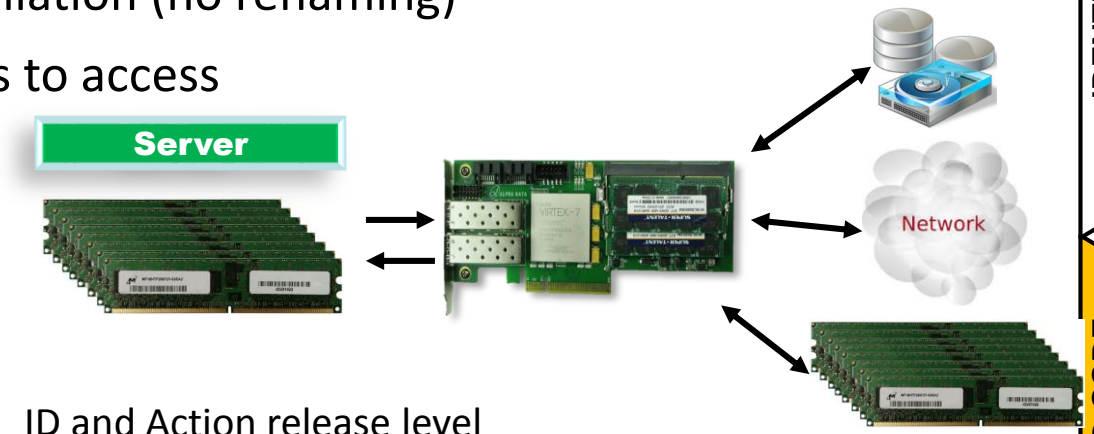
SNAP files organization:

SNAP
+ actions
+ hls_helloworld
+ sw *(cpu)*
- snap_helloworld.c
- action_lowercase.c
+hw *(fpga)*
- action_uppercase.cpp
+ include
- action_changecase.h

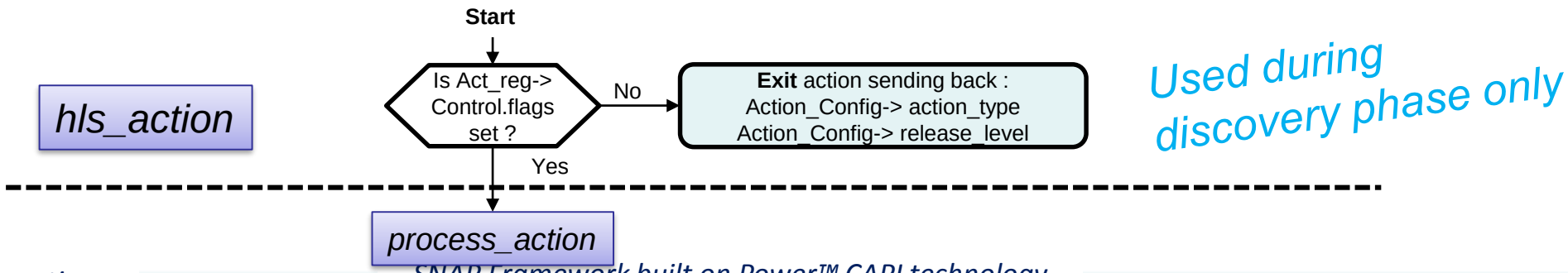
Working directory : hls_helloworld/hw

hls_action : ports declaration and discovery answer

- hls_action is the “top” mandatory name for SNAP compilation (no renaming)
- **Ports declaration**: defines which resources action needs to access
  - **din_gmem** : read access to the server memory
  - **dout_gmem** : write access to the server memory
  - **d_ddrmem**: read/write access to the card embedded DDR



- **Registers mapping**
  - **ActionConfig** registers : Read Only registers containing Action_ID and Action release level
  - **act_reg** registers : Read/Write registers containing all parameters exchanged between application and action
- **Discovery phase answer**: before first use, SNAP need to “discover” the available actions on different cards of the server. Every action need to answer his type and release level if requested by the SNAP job manager by the program ***snap_maint*** (available in snap/software/tools).



HLS compilation : “converts” C code to RTL code (for FPGA)

➔ Compile the “hardware” action alone

```
➢ cd snap/actions/hls_helloworld/hw
➢ make
```

Checking for critical warnings during HLS synthesis ... OK
Checking for reserved MMIO area during HLS synthesis ... OK

SNAP files organization:

SNAP
+ actions
+ hls_helloworld
+ sw *(cpu)*
- snap_helloworld.c
- action_lowercase.c
+hw *(fpga)*
- action_uppercase.cpp
+ include
- action_changecase.h

IDENTIFY
MODIFIED CODE
EXECUTION

Execution Phase
2nd STEP:
Create the "hardware" action code from the "software" action

1) **Identify and understand** what to modify
2) **Discover** the modified code
3) **Compile** and **execute** the code on CPU
    1) Set the environment
    2) Build the simulation model
    3) Execute the hardware action

IDENTIFY
MODIFIED CODE
EXECUTION

Set the SNAP environment

Define the environment (if not done yet)

```
➢ cd snap
➢ make clean_config (optional)
➢ make snap_config
```

This will :

- *Clean environment*
- *Launch a menu to let you set the environment you want to work on.*
  - ➔ *Select AD8K5 card*
  - ➔ *Select Hello World action*

Typical display:

```
hdclv009 hardware$ make clean_config
        [CLEAN ENVIRONMENT...] start Mon Sep  4 14:11:41 CEST 2017
                        vivado project
                        IPs
                        sim files
                        log files
                        action
        [CLEAN ENVIRONMENT...] done Mon Sep  4 14:11:42 CEST 2017

hdclv009 hardware$ make snap_config
menuconfig: Setting up SNAP configuration
…/…
# configuration written to .config
Generating SNAP config files ../.snap_config, ../.snap_config.sh and
../.snap_config.cflags ...
Generation of SNAP config files done.
…/…
SNAP config done
```

This command ***make clean_config*** will be executed every time you change the card type

IDENTIFY | MODIFIED CODE | EXECUTION

Build the action "hardware model" for simulation

Build the simulation model:

```
➢ cd snap
➢ make model / make sim
```

#2mins

This will :

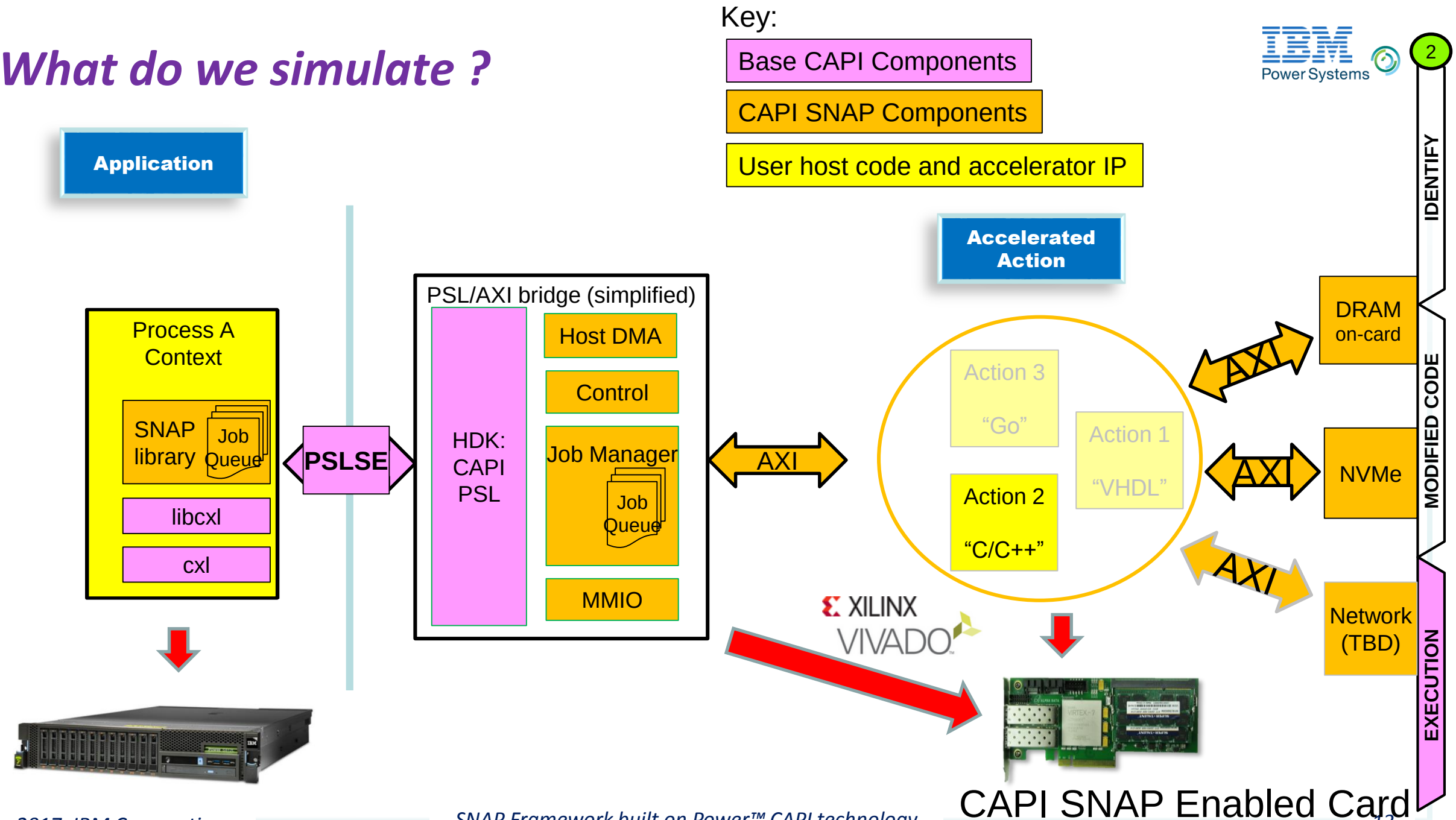
- *Compile PSLSE (PSL simulation engine)*
- *Compile Software application*
- *Configure the SNAP cores*
- *Build the simulation model*

Typical display:

```
hdclv009 hardware$ make model
    [CREATE_FRAMEWORK....] start
    [CREATE_FRAMEWORK....] done
    [CREATE_ENVIRONMENT..] done Mon Sep  4 14:13:00 CEST 2017
    [PATCH...............] SNAP version and build date registers
    [COMPILE PSLSE ......] start Mon Sep  4 14:16:48 CEST 2017
    [COMPILE PSLSE ......] done Mon Sep  4 14:16:48 CEST 2017
    [COMPILE SOFTWARE....] start Mon Sep  4 14:16:48 CEST 2017
    [COMPILE SOFTWARE....] done Mon Sep  4 14:16:48 CEST 2017
    [COMPILE ACTION SW...] start Mon Sep  4 14:16:48 CEST 2017
    [COMPILE ACTION SW...] done Mon Sep  4 14:16:48 CEST 2017
    [CONFIG SNAP CORE....] start Mon Sep  4 14:16:48 CEST 2017
    [CONFIG SNAP CORE....] done Mon Sep  4 14:16:48 CEST 2017
    [PATCH...............] SNAP version and build date registers
    [BUILD xsim..........] start Mon Sep  4 14:16:48 CEST 2017
                          export simulation for version=2016.4
    [BUILD xsim..........] done  Mon Sep  4 14:18:09 CEST 2017
```

IDENTIFY
MODIFIED CODE
EXECUTION

What do we simulate ?



Run the simulation : starting by the discovery phase

```
➢  make sim
or cd hardware/sim + ./run_sim
```

This will open a terminal where you can execute all commands you need:

snap_maint ➔ *basic settings done by the SNAP job manager to configure the card*

```
➢  snap_maint -v
```

snap_maint –v

```
INFO:Connecting to host port 16384
SNAP on AD8K5 Card, NVME disabled, 0 MB SRAM available.
SNAP FPGA Release: v1.1.0 Distance: 167 GIT: 0x6a4c1d1b
SNAP FPGA Build (Y/M/D): 2017/09/18 Time (H:M): 18:34
SNAP FPGA CIR Master: 1 My ID: 0
SNAP FPGA Up Time: 0 sec
   0 Max AT: 1 Found AT: 0x10141008 --> Assign Short AT: 0
     0     0x10141008     0x00000021  IBM HLS Hello World

INFO:detach response from from pslse
```

PSL Simulation engine results

IDENTIFY
MODIFIED CODE
EXECUTION

Run the simulation : hardware action will be executed

Staying in the simulator window, run now the application calling the hardware action

(remember that SNAP_CONFIG=FPGA meaning running the hardware is by default)

```
➢  cat /tmp/t1
If file doesn't exist, create it:
➢  vi /tmp/t1
    Hello World. I love this new experience with SNAP.

➢  $ACTION_ROOT/sw/snap_helloworld -v -i /tmp/t1 -o /tmp/t2
➢  cat /tmp/t1
➢  cat /tmp/t2
```

Results next slides

snap_helloworld *:*
-v(v) : verbose mode
(-t 60 : timeout in secs)
(-N : disable Interrupts – polling mode)
-i /tmp/t1 : input file
-o /tmp/t2 : output file

2
IDENTIFY
MODIFIED CODE
EXECUTION

Running hls_helloworld action with the simulator

```
$ ACTION_ROOT/sw/snap_helloworld -i /tmp/t1 -o /tmp/t2
helloworld -v -i/tmp/t1 -o/tmp/t2
reading input data 51 bytes from /tmp/t1
PARAMETERS:
  input:        /tmp/t1
  output:       /tmp/t2
  type_in:      0 HOST_DRAM
  addr_in:      0000000001d17000
  type_out:     0 HOST_DRAM
  addr_out:     0000000001d18000
  size_in/out:  00000033
INFO:Connecting to host 'hdclv010.boeblingen.de.ibm.com' port 16384
  prepare helloworld job of 32 bytes size
writing output data 0x1d18000 51 bytes to /tmp/t2
RETC=102
SNAP helloworld took 4364014 usec
INFO:detach response from from pslse
$ cat /tmp/t1
Hello World. I love this new experience with SNAP.

$ cat /tmp/t2
HELLO WORLD. I LOVE THIS NEW EXPERIENCE WITH SNAP.
```

Caution: Do not take the reported time in account since software is counting the time of the simulator!

Check how to switch to the CPU execution

```
➢   SNAP_CONFIG=CPU $ACTION_ROOT/sw/snap_helloworld  -i /tmp/t1 -o /tmp/t2
cat /tmp/t2
hello world. i love this new experience with snap.
```

Debugging ?
➔ try ***SNAP_TRACE=0xF***

IDENTIFY
MODIFIED CODE
EXECUTION

Step 2: what you need to remember

IBM Power Systems

2

IDENTIFY

MODIFIED CODE

EXECUTION

2nd step: **We know that the code generated for the FPGA works ok (on CPU)**

Steps	Effect of the command	Files used	Working directory	Command used	Target
2 simulate hw action	***convert C hw action to RTL***	action_uppercase.cpp	~/snap/actions/hls_helloworld/hw	**make**	x86
	compile all hw design for simulation	action_uppercase.cpp	~/snap	**make model or make sim**	
	simulate hw action	snap_helloworld.c + action_uppercase.cpp + /tmp/t1	~/snap/hardware/sim	**./run_sim** $SNAP_ROOT/software/tools/**snap_maint -v** $ACTION_ROOT/sw/**snap_helloworld** -i/tmp/t1 -o/tmp/t2	

main files:

- Application : snap_helloworld.c
  - software action : action_lowercase.c
  - hardware action : action_uppercase.cpp
  - *common header : action_changecase.h*

SNAP files organization:

SNAP
+ actions
+ hls_helloworld
+ sw *(cpu)*
- snap_helloworld.c
- action_lowercase.c
+hw *(fpga)*
- action_uppercase.cpp
+ include
- action_changecase.h

Questions ?