OmpSs@FPGA

Tutorial:

Installation and Hands-on

OmpSs@FPGA BSC team

November 4th, 2018

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Outline

- OmpSs@FPGA Ecosystem
  - Zynq Support + Hands on Objectives
  - Requirenments and Limitations
  - Computer and SD setup
- Cross Compilation and Execution
- Porting of Applications
  - First: OmpSs@SMP
  - Second: OmpSs@FPGA

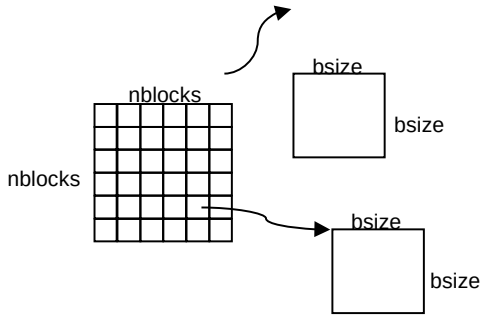
Outline

- **OmpSs@FPGA Ecosystem**
  - Zynq Support + Hands on Objectives
  - Requirenments and Limitations
  - Computer and SD setup
- **Cross Compilation and Execution**
- **Porting of Applications**
  - First: OmpSs@SMP
  - Second: OmpSs@FPGA

OmpSs@FPGA:

Easy to program: MxM

```
#pragma omp target device(fpga) copy_deps num_instances(2)
#pragma omp task in([BS]a, [BS]b) inout([BS]c)
void matrix_multiply(T a[BS][BS], T b[BS][BS], T c[BS][BS]);
```



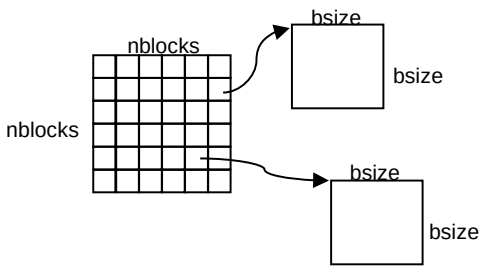

```
…
For (i_b=0; i_b<NB_I; i_b++)
  for (j_b=0; j_b<NB_J; j_b++)
    for (k_b=0; k_b<NB_K; k_b++)
      matrix_multiply(AA[i_b][k_b], BB[k_b][j_b], CC[i_b][j_b]);
…
#pragma omp taskwait
// Or
// other tasks depending on input output c of matrix multiply task
```

OmpSs@FPGA:

Easy to program: MxM

```
#pragma omp target device(fpga) copy_deps num_instances(2)
#pragma omp task in([BS]a, [BS]b) inout([BS]c)
void matrix_multiply(T a[BS][BS], T b[BS][BS], T c[BS][BS]);

#define BS 128
void matrix_multiply(float a[BS][BS], float b[BS][BS],float c[BS][BS])
{
#pragma HLS inline
  int const FACTOR = BS/2;
#pragma HLS array_partition variable=a block factor=FACTOR dim=2
#pragma HLS array_partition variable=b block factor=FACTOR dim=1
  // matrix multiplication of a A*B matrix
  for (int ia = 0; ia < BS; ++ia)
    for (int ib = 0; ib < BS; ++ib) {
#pragma HLS PIPELINE II=1
      float sum = 0;
      for (int id = 0; id < BS; ++id)
        sum += a[ia][id] * b[id][ib];
      c[ia][ib] += sum;
    }
}
```



OmpSs@FPGA
Current Limitations

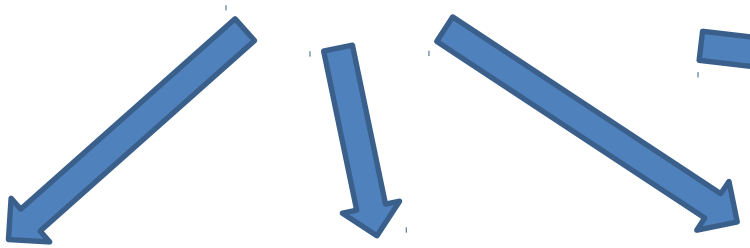
- ***Limitations***
  - Global variables and constants should be within an include with name extension .fpga or .fpga.h
  - Only C/C++ support, no fortran
  - Accelerator task should be call from the same source file
  - Only function declarations can be annotated as a fpga task
  - All arguments of the function must be of pointer type
  - Function memcpy should be avoided from HLS code... or it would be interpreted as Vivado HLS memcpy
  - Name and path … can not contain certain characters (@, etc.)
  - FPGA tasks cannot perform system calls, neither calls to the Nanos++ API
  - FPGA tasks cannot create other tasks

OmpSs@FPGA:

Easy to program: MxM

```
#pragma omp target device(fpga) copy_deps num_instances(2)
#pragma omp task in([BS]a, [BS]b) inout([BS]c)
void matrix_multiply(T a[BS][BS], T b[BS][BS], T c[BS][BS]);
```

OmpSs Application

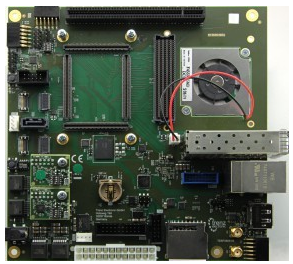


Zynq ZU102

Zedboard and Xilinx devkits
XC702 and XC706

2x Cortex-A9 cores + FPGA
32-bit platforms

SECO AXIOM Board
Zynq U+ XCZU9EG-ES2



Trenz Electronics Zynq U+
TE0808 XCZU9EG-ES1

4x Cortex-A53 cores + FPGA
64-bit platforms

OmpSs@FPGA:

Easy to program: MxM

```
#pragma omp target device(fpga) copy_deps num_instances(2)
#pragma omp task in([BS]a, [BS]b) inout([BS]c)
void matrix_multiply(T a[BS][BS], T b[BS][BS], T c[BS][BS]);
```

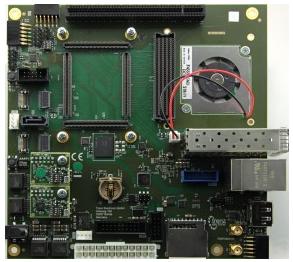
- OBJECTIVES:
- Do hands-on with small examples – Faster to test
- Load bitstream while running the system
  - - Linux: Ubuntu linaro + petalinux

OmpSs Application

Zedboard
2x Cortex-A9 cores + FPGA
32-bit platforms

SECO AXIOM Board
Zynq U+ XCZU9EG-ES2

Zynq ZU102



Trenz Electronics Zynq U+
TE0808 XCZU9EG-ES1

4x Cortex-A53 cores + FPGA
64-bit platforms

OmpSs@FPGA:

Easy to program: MxM

```
#pragma omp target device(fpga) copy_deps num_instances(2)
#pragma omp task in([BS]a, [BS]b) inout([BS]c)
void matrix_multiply(T a[BS][BS], T b[BS][BS], T c[BS][BS]);
```

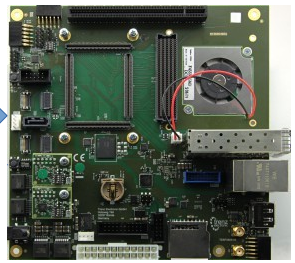
OmpSs Application

- OBJECTIVES
- Test you can cross-compile for a Trenz
- Linux: Ubuntu linaro + petalinux
  - So far: no possible to load a bistream while running…
- Show you how to generate the boot: BOOT.bin + image.ub

Zedboard and Xilinx devkits
XC702 and XC706

2x Cortex-A9 cores + FPGA
32-bit platforms

SECO AXIOM Board
Zynq U+ XCZU9EG-ES2



Trenz Electronics Zynq U+
TE0808 XCZU9EG-ES1
4x Cortex-A53 cores + FPGA
64-bit platforms

OmpSs@FPGA:
Easy to Compile/Run in Parallel

- **Compile:** ***fpgacc mercurium alias***
  - Without instrumentation
    - ***cross-compiler-fpgacc –ompss -o program program.c***
  - With instrumentation
    - ***cross-compiler-fpgacc – ompss –instrument -o program program.c***
- **Run**: ***Heterogeneous Execution***
  - Without instrumentation
    - ***./program input_args***
  - With instrumentation
    - ***NX_ARGS="--instrumentation=extrae" ./program input_args***

Computer and SD setup

- ***Computer Cross Compilation***
  - ***Docker*** e.g., https://download.docker.com/linux/static/stable/x86_64/
- ***SD structure and Description***
- ***Petalinux 2016.3***

OmpSs@FPGA:
Current Requirements

- ***Software Requirenments***
  - ***Vivado Full Design Suite***
  - ***Mercurium*** @ FPGA (Computer: cross-compiling)
    - Back-end Cross compiler (***arm-linux-gnueabihf-, aarch64-linux-gnu-***) gcc,g++,gfortran
    - ***autoVivado*** 2016.3 or 2017.3
  - ***Nanox*** @ FPGA (/opt/install-arm[64]/nanox/) (SD and Computer: the same)
  - ***libxtasks, libxdma*** @ FPGA (/opt/install-arm[64]/libxtasks/, /opt/install-arm[64]/libxdma)) (SD and Computer: the same)
  - ***Extrae*** @ FPGA (/opt/install-arm[64]/extrae) (SD and Computer: the same)
  - ***taskmanager and xdma*** drivers (SD any directory)
- **Hardware Requirenments:**
  - Zynq 7000/Zynq U+

OmpSs@FPGA: Cross Compiling to Zynq

- ***Docker image + Makefile prepared***
  - ***Contains:***
    - ***Mercurium (+ autovivado ← To be updated)***
    - ***Nanox***
    - ***Libxtask, libxdma***
    - ***Extrae, Paraver***
    - ***Examples: example-board***
      - ***With environment-auto.sh***
    - ***Petalinux-prebuilds: petalinux-board***
    - ***Instrumentation files: extrae.xml and .cfg file for paraver***
    - ***ARM gcc cross-compilers***
  - ***Does not contains:***
    - ***Petalinux, Vivado, Vivado_HLS***

OmpSs@FPGA:
Docker Makefile changes

- ***WARNING: Petalinux installation directory should be seen from any of the below directories if you want to generate the BOOT.bin and image.ub during the compilation.***
- ***Changes in target run:***
  - ***/home/<username>/ by your home directory in your system***
  - ***/opt/Xilinx/ or /Xilinx/ to the directory where you will find Sdx, SDK or Vivado software in your system***

    Warning: later, in the environment-auto.sh you may need to change the directory of the vivado and vivado_hls where it should be done the source of the settings.sh
  - ***/media/ to the directory where you will find your usb media devices***

OmpSs@FPGA Docker Container

Docker image and Makefile provided

- ***TO RUN JUST ONCE:***
  - ***Uncompress the image provided (***gunzip ompssatfpga.img.gz***)***
  - ***Load the image*** (sudo make load)
  - **docker image load -i ompssatfpga.img**
    - **OmpSs@FPGA ecosystem, example-zedboard/trenz/zcu102**
  - **Run container** (sudo make run) : It creates, starts and attaches a container
  - **docker run -it -v … -e DISPLAY=$(DISPLAY) --network host ompssatfpga/training-2018**
    - **-v → Vivado binaries, user home at host, other directories**
- ***…***
- ***exit***
- ***TO CONTINUE WORKING with container*** (sudo make attach)
  - ***It does (supposing you only have one container running):***
    - ***docker ps -a ; docker start container_id; docker attach container_id***

OmpSs@FPGA:
Container requires some updates

- ***Autovivado***
  - ***Replace autovivado at home directory with the autovivado.tar.gz provided***
- ***BOOT.bin generation: Modify environment-auto.sh***

```
export PETALINUX_BUILD=/home/ubuntu/petalinux-trenz/trenz_petalinux/
export PETALINUX_INSTALLATION=TOFILL
source $PETALINUX_INSTALLATION/settings.sh
```

  - ***Should be replaced by:***

```
export PETALINUX_BUILD=/home/ubuntu/petalinux-trenz/trenz_petalinux/
export PETALINUX_INSTALL=to_fill_with_path_of_petalinux_installation/
```

- ***Access to Vivado: Modify environment-auto.sh***
  - ***export PATH=$PATH:/...:/opt/Xilinx/Sdx/2016.3/Vivado/bin/:/opt/Xilinx/Sdx/2016.3/Vivado_HLS/bin/***
- ***Instrumentation files: copy task_name_and_fpga.cfg to Instrumentation directory***

SD Structure and Description

- ***Copy SD image to your SD card***
  - Zynq-7000
    - sudo umount /dev/mmcblk0
    - sudo dd if=zedboard.img of=/dev/mmcblk0 bs=4M
  - ZU
    - sudo umount /dev/mmcblk0
    - sudo dd if=trenz.img of=/dev/mmcblk0 bs=4M

Outline

- OmpSs@FPGA Ecosystem
  - Zynq Support + Hands on Objectives
  - Requirenments and Limitations
  - Computer and SD setup
- Cross Compilation and Execution
- OmpSs@FPGA Advanced

Cross Compiling and Execution

- *Docker Cross Compilation*
- *Boot and application setup*
- *Execute on Board*

Cross Compiling

- ***Docker (make attach) Cross Compilation***
  - We have provided you with a example directory with a sample vector mult code to compile within the docker container:
    - For zedboard:
      - cd example-zedboard
    - For trenz:
      - cd example-trenz
    - For zcu102:
      - cd example-zcu102
  - Edit Makefile to see what is inside

Complete Environment (Zynq-7000)

- ***Example of environment-auto.sh***
  - export CROSS_COMPILE=arm-linux-gnueabihf-
  - export PATH=$PATH:/home/ubuntu/autoVivado/scripts
  - export PATH=$PATH:/opt/mcxx-arm/bin/:/pathto/Vivado/2016.3/bin/:/pathto/Vivado_HLS/2016.3/bin/
  - export PETALINUX_BUILD=/home/ubuntu/petalinux-zedboard/Avnet-Digilent-ZedBoard-2016.3
  - export PETALINUX_INSTALL=...

- ***Then run:***
  - *source environment-auto.sh*

Complete Environment (ZU+ trenz)

- ***Example of environment-auto.sh***

  export CROSS_COMPILE=aarch64-linx-gnu-

  export PATH=/home/ubuntu/autoVivado/scripts/:$PATH

  export PATH=$PATH:/opt/mcxx-arm64/bin/:/pathto/Vivado/2016.3/bin/:/pathto/Vivado_HLS/2016.3/bin/

  export PETALINUX_BUILD=/home/ubuntu/petalinux-trenz/trenz_petalinux/

  export PETALINUX_INSTALL=TOFILL

- ***Then run:***
  - *source environment-auto.sh*

Complete Environment (ZU+ zcu102)

- ***Example of environment-auto.sh***

  export CROSS_COMPILE=aarch64-linx-gnu-

  export PATH=/home/ubuntu/autoVivado/scripts/:$PATH

  export PATH=$PATH:/opt/mcxx-arm64/bin/:/pathto/Vivado/2017.3/bin/:/pathto/Vivado_HLS/2017.3/bin/

  export PETALINUX_BUILD=/home/ubuntu/petalinux-zcu102/euroexa-project/

  export PETALINUX_INSTALL=TOFILL

- ***Then run:***
  - *source environment-auto.sh*

Cross Compiling Details

- Few Details on the steps to compile and run
  - Compile: use fpgacc
    - cross-compile-fpgacc --ompss –o program program.c
    - cross-compile-fpgacc --ompss --instrumentation –o program program.c
    - *Details:*
      - *Compiling options:*

        --variable=bitstream_generation:ON
      - *Linking options:*

        *--Wf,"--board=$(BOARD_NAME),--clock=100,--hardware_instrumentation,--task_manager,--to_step=boot"**

        --Wf,"-v,--name=vivado_project_name,--dir=$(VIVADO_WORKSPACE)"

* --to_step=boot is not necessary for the zynq7000 (zedboard)

Cross Compiling Details

```
usage: autoVivado [-h] -b BOARD -n NAME [-c CLOCK] [-d DIR]
                  [--from_step FROM_STEP] [--to_step TO_STEP]
                  [--hardware_instrumentation] [--task_manager] [--task_batch]
                  [--IP_cache_location IP_CACHE_LOCATION]
                  [--disable_IP_caching] [--enable_DMA]
                  [--intercon_opt INTERCON_OPT] [-v]
                  [--disable_utilization_check]

optional arguments:
  -b BOARD, --board BOARD
                        board model. Supported boards for Vivado 2016.3:
                        axiom, axiom_nic, ikergune, trenz, zedboard, zynq702, zynq706
  -n NAME, --name NAME  project name
  -c CLOCK, --clock CLOCK                    FPGA clock frequency in MHz (def: '100')
  -d DIR, --dir DIR     path where the project directory tree will be created (def: './')
  --from_step FROM_STEP
                        initial compilation step. Compilation steps:
                        HLS, design, synthesis, implementation, bitstream, boot (def: 'HLS')
  --to_step TO_STEP     final compilation step. Compilation steps:
                        HLS, design, synthesis, implementation, bitstream, boot (def: 'bitstream')
  --hardware_instrumentation
                        adds hardware support for hardware instrumentation
  --task_manager        adds hardware support for Task Manager
  --task_batch          adds hardware support for Task Batch
  --IP_cache_location IP_CACHE_LOCATION
                        path where the IP cache will be located (def: '<autoVivado>/Vivado/IP_cache/')
  --disable_IP_caching  disables IP caching. Significantly increases compilation time
  --enable_DMA          enables DMAs for synchronous communication. Increases FPGA utilization
  --intercon_opt INTERCON_OPT
                        AXI interconnect optimization strategy: Minimize 'area' or maximize 'performance'  (def: 'area')
  -v, --verbose         prints Vivado messages
  --disable_utilization_check
                        disables resources utilization check during HLS generation
```

Cross Compiling for ZU: Let's go

- ***Run:*** source environment-auto.sh
- ***Edit Makefile and see the targets and compilation flags***
- ***Try to compile***
  - make vector_mult_smp_blocking_multiple_acc_trenz
  - Or
  - make vector_mult_smp_blocking_multiple_acc_zcu102

Cross Compiling for Zynq-7000: Let's go

- ***Run:*** source environment-auto.sh
- ***Edit Makefile and see the targets and compilation flags***
  - The same :-)
- ***Try to compile:***

1) make vector_mult_smp or
2) make vector_mult_smp_blocking_multiple_acc_zedboard

BOOT and Application setup

- ***Zynq 7000***
  - **BOOT:**
    - Nothing to be done, already setup
    - We can donwload bitstream while running the linux system
  - **Application**
    - Copy bitstream, SMP binary and config file (xtasks.config)
- ***Zynq Ultrascale+***
  - ***BOOT:***
    - ***Petalinux 2016.3 (trenz bsp) or 2017.3***
      - Installation and BOOT.bin and image.ub generation
      - Copy them to boot partition (generated in the vivado_project)
  - **Application:**
    - Bistream is already within the BOOT.bin. Copy SMP binary and config file (xtasks.config)

Application setup and connect to the Board

- ***Copy (cp or scp) to /home/ubuntu/examples/ (rootfs partition)***
  - Program SMP OmpSs binary
  - vivado_project/Program.xtasks.config : indicates the number of accelerators and instances
  - Case Zynq-7000: vivado_project/Program_vivado/Program.bin bitstream
- ***Insert SD to your board and Boot it***

- ***Connect to:***
  - **First option: SSH**:
    - ssh -X ubuntu@192.168.1.10
    - (password: ubuntu)
  - ***Minicom:*** minicom -D /dev/ttyACM0
    - (ACM0, this is zero, or ttyUSB0) + login ubuntu (pass:ubuntu)

Boot setup for ZU

- ***Zynq Ultrascale+***
  - ***Edit environment-auto.sh***
    - export PETALINUX_INSTALL=path to the petalinux 2016.3 installation
    - PETALINUX_BUILD points to an configured petalinux kernel for the target board
  - **Makefile**
    - Warning:
      - Add -L/opt/install-arm/libxtasks/lib -lxtasks to Makefile
    - --to_step especify that we want to generate the BOOT.bin and image.ub
  - Warning: /home/ubuntu/petalinux-directory/directory (if necessary)
    - petalinux-build -x distclean
    - petalinux-config // Press exit
    - petalinux-config -c kernel // Press exit

Execution Details

- Few details on the steps to compile and run
  - Load (insmod) drivers
    - sudo insmod */opt/modules/xdma.ko*
    - *sudo insmod /opt/modules/taskmanager.ko*
  - Loading Bitstream: it is already loaded in the FPGA, However….
    - Zynq 7000 : Can be loaded in hot
    - Trenz: Generate BOOT.bin and image.ub
    - ZCU102: Work in progress… in hot too
  - Then Run the program ….

Execution Details

- Few details on the steps to compile and run
  - Zynq: Run Program: ./program <input arguments>
  - Zynq: Run w/ instrumentation + Paraver
    - export EXTRAE_CONFIG_FILE=extrae.xml
    - NX_ARGS="--instrumentation=extrae" ./program <input>
    - *Details:*
      - *export EXTRAE_CONFIG_FILE="extrae.xml"*
  - Container: Run wxparaver file.prv
    - Copy .prv, .pcf, .row from Zynq to Container
    - Configuration file:
      - Instrumentation directory at docker: task_name_and_fpga.cfg

OmpSs@FPGA
Tutorial:
Installation and Hands-on

OmpSs@FPGA BSC team

November 2018

Petalinux 2016.3 installation

- Zynq Ultrascale+ (from Trenz bsp)

Petalinux 2016.3 installation (I/III)

- Run:
  - ./petalinux-v2016.3-final-installer.run
    - Be sure that you are NOT in a console with the Xilinx environment
  - cd petalinux-directory
  - source settings.sh

- Copy the bsp file in USB/ZU/bsp/ to any place in your computer and then generate the kernel
  - There are two stages

    1) Steps run just once and are useful to configure the baseline kernels files

    2) [Temporally] Steps done for each new application to be accelerated

Petalinux 2016.3 installation (II/III)

- Run just ONCE:

1) Unpack the bsp provided in USB/ZU/bsp/
   - petalinux-create -t project -s <path to petalinux bsp>
   - cd petalinux-bsp-project
2) petalinux-config
3) Activate and increase CMA
   - petalinux-config -c kernel

     Then Device_Drivers/*Generic Driver Options*/ Size in Megabytes → 512
4) Build kernel
   - petalinux-build

Petalinux 2016.3 installation (III/III)

- Run each application is accelerated (**after each new hw**): :

1) Import hdf into petalinux project
   - petalinux-config --get-hw-description <path to application hdf file>
2) Add missing nodes to device tree
   - Edit the file ./subsystems/linux/configs/device-tree/pl.dtsi according to next sections

```
misc_clk_0: misc_clk_0 {
  compatible = "fixed-clock";
  #clock-cells = <0>;
  clock-frequency = <100>;
};
```

```
xdma@0 {
  compatible = "xdma,xdma_acc";
  instrument-timer = <&Hardware_Instrumentation_BRAM_Ctrl>;
};
```

3) Build the linux system
   - petalinux-build
4) Create boot.bin file
   - petalinux-package --force --boot --fsbl images/linux/zynqmp_fsbl.elf --fpga <path to application bit file> --u-boot images/linux/u-boot.elf
   - cp BOOT.BIN images/linux/image.ub <path to boot partition>

Petalinux 2016.3 installation

ZCU102 (from scratch)

Petalinux 2016.3 installation (I/IV)

- Run:
  - ./petalinux-v2016.3-final-installer.run
    - Be sure that you are NOT in a console with the Xilinx environment
  - cd petalinux-directory
  - source settings.sh

Petalinux 2016.3 installation (II/IV)

- Run just ONCE:

1) Create project
   - petalinux-create -t project --template zynqMP --name project_name
   - cd project_name
   - Keep PWD for step 3
2) Copy hdf to a directory, for instance:
   - cd hw-description/ ; mkdir project_name; cd project_name
   - cp path-to-hdf/project_name.hdf .
3) Kernel configuration from hdf
   - petalinux-config --get-hw-description -p path-Step-1
     Only change: Image Packaging Configuration:
     choose SD in : Root filesystem type (SD card) ---> SD
     … unset: [ ] Copy final images to tftpboot
4) Activate and increase CMA
   - cd ../../
   - petalinux-config -c kernel
     Then Device_Drivers/*Generic Driver Options*/ Size in Megabytes → 512
5) Build kernel
   - petalinux-build

Petalinux 2016.3 installation (III/IV)

- Run each application is accelerated (**after each new hw**): :

1) Import hdf into petalinux project
   - petalinux-config --get-hw-description <path to application hdf file>
2) Add missing nodes to device tree
   - Edit file ./subsystems/linux/configs/device-tree/pl.dtsi according to next sections, after “Task_Manager_Task_Manager_rst”

```
misc_clk_0: misc_clk_0 {
  compatible = "fixed-clock";
  #clock-cells = <0>;
  clock-frequency = <100>;
};
```

```
xdma@0 {
  compatible = "xdma,xdma_acc";
  instrument-timer = <&Hardware_Instrumentation_BRAM_Ctrl>;
};
```

   - Edit file ./subsystems/linux/configs/device-tree/system-top.dts, after /include/ "zynqmp-clk.dtsi"

```
&gem3 {
 phy-handle = <&phyc>;
 phyc: phy@c {
            reg = <0xc>;
            ti,rx-internal-delay = <0x8>;
            ti,tx-internal-delay = <0xa>;
            ti,fifo-depth = <0x1>;
 };
 /* Cleanup from RevA */
 /delete-node/ phy@21;
};
```

Petalinux 2016.3 installation (IV/IV)

- Run each application is accelerated (**after each new hw**): :

1) Build the linux system
   - petalinux-build
2) Create boot.bin file
   - petalinux-package --force --boot --fsbl images/linux/zynqmp_fsbl.elf --fpga <path to application bit file> --u-boot images/linux/u-boot.elf
   - cp BOOT.BIN images/linux/image.ub <path to boot partition>
3) Copy BOOT.BIN + image.ub to BOOT partition