PLC2

Vitis from Zero to Hero

PLC2 online – 06 March 2020

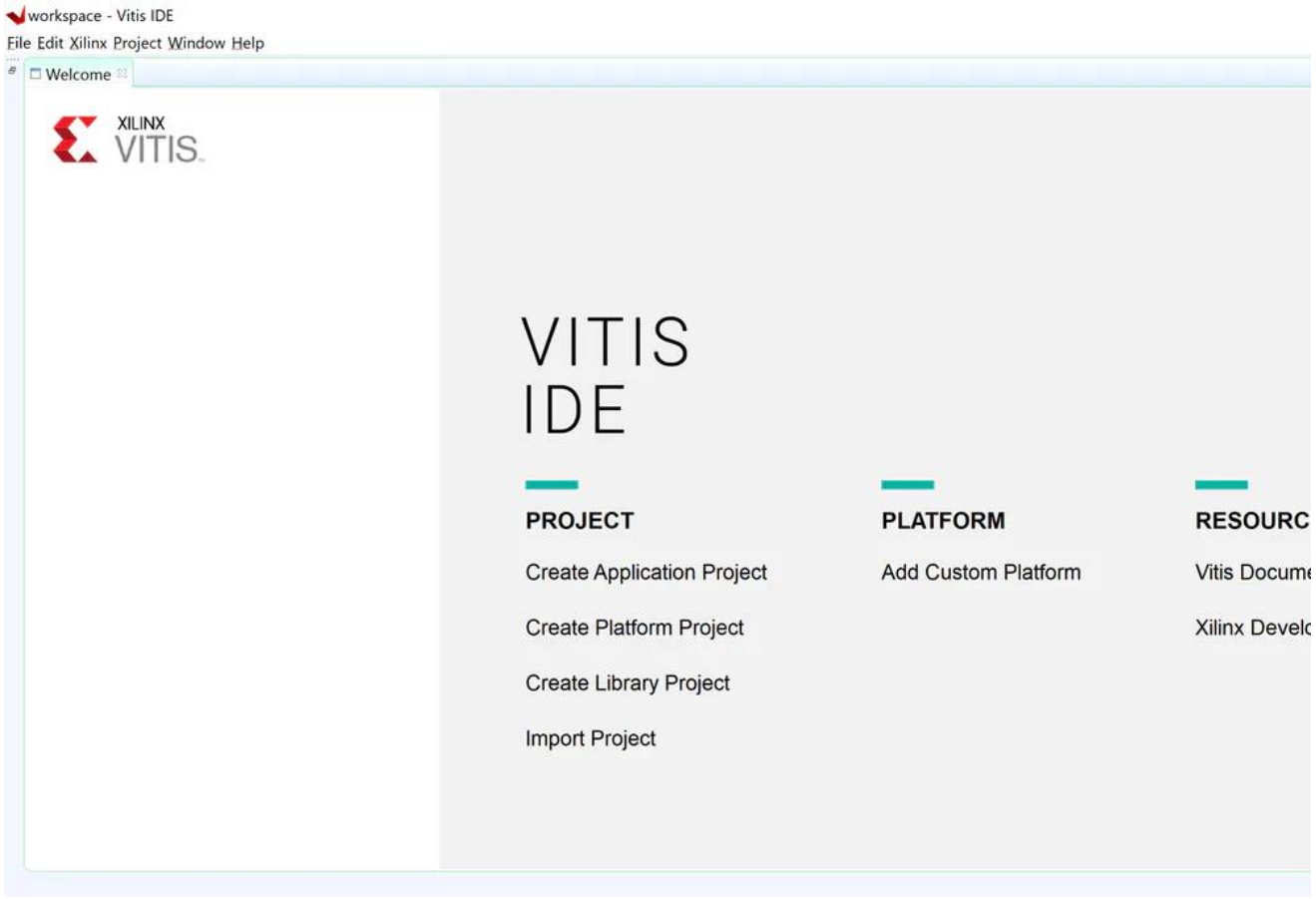
What is Vitis?

- Xilinx unified software development environment
- Enables development of embedded, accelerated and AI solutions on Xilinx devices
- Works hand in hand with Vivado and Petalinux
- Works on Linux and Win 10
  - Linux only supported for acceleration flow

Key concepts of Vitis

Slightly different concepts from SDK

- Platform Project
  - HW & SW resources e.g. BSP, FSBL, HW Spec
  - Grouped as domains
    - BSP, OS etc.
- System Project
  - Actual applications which are created
- Debugging on target and on QEMU
  - System Debugger



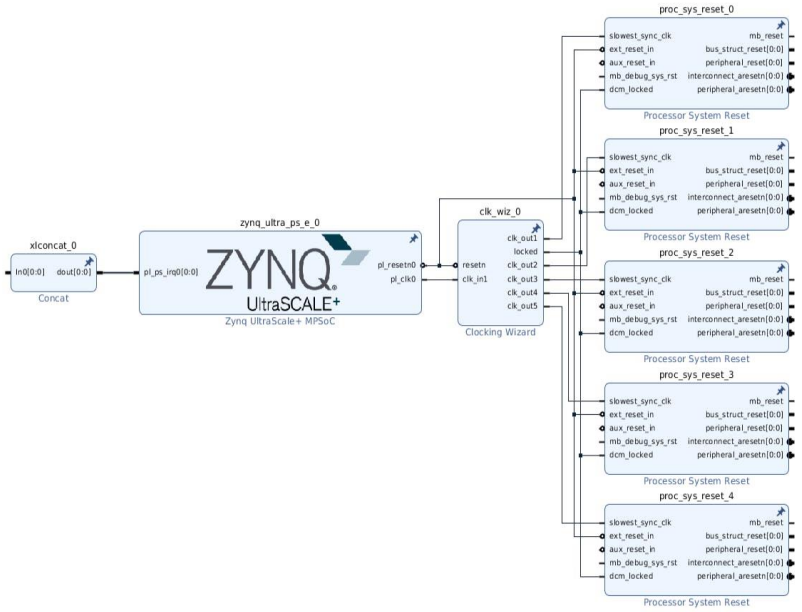
Vitis Flows

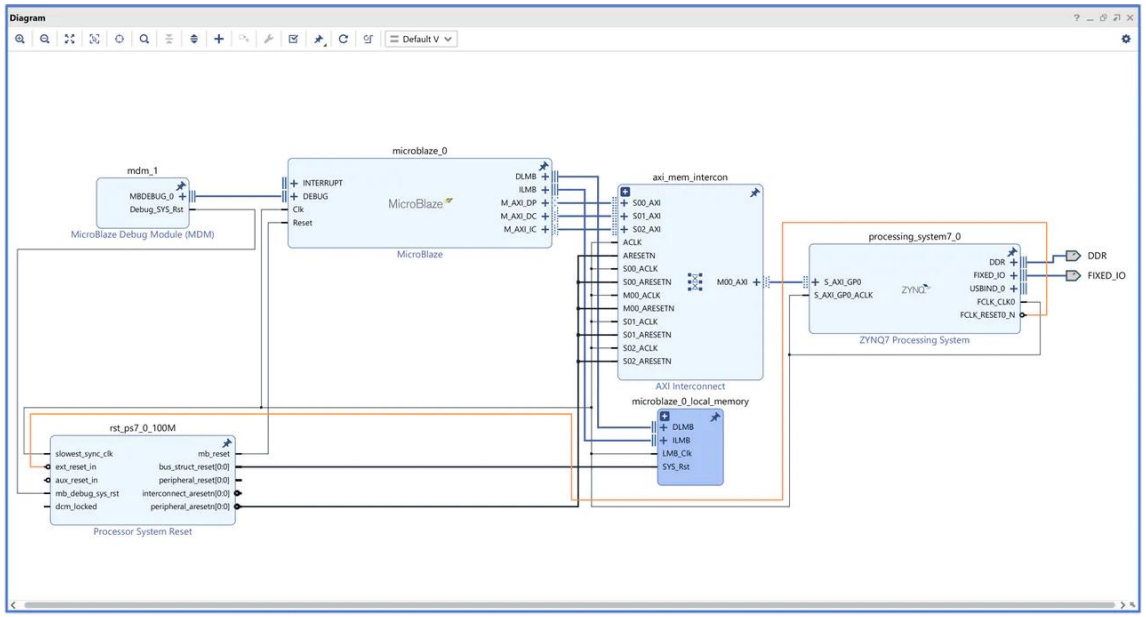
Accelerated

- Leverages OpenCL to accelerate functions into the programmable logic
- Uses Xilinx Run Time (XRT)
- Petalinux Only
- Requires specific configuration in Vivado and Petalinux

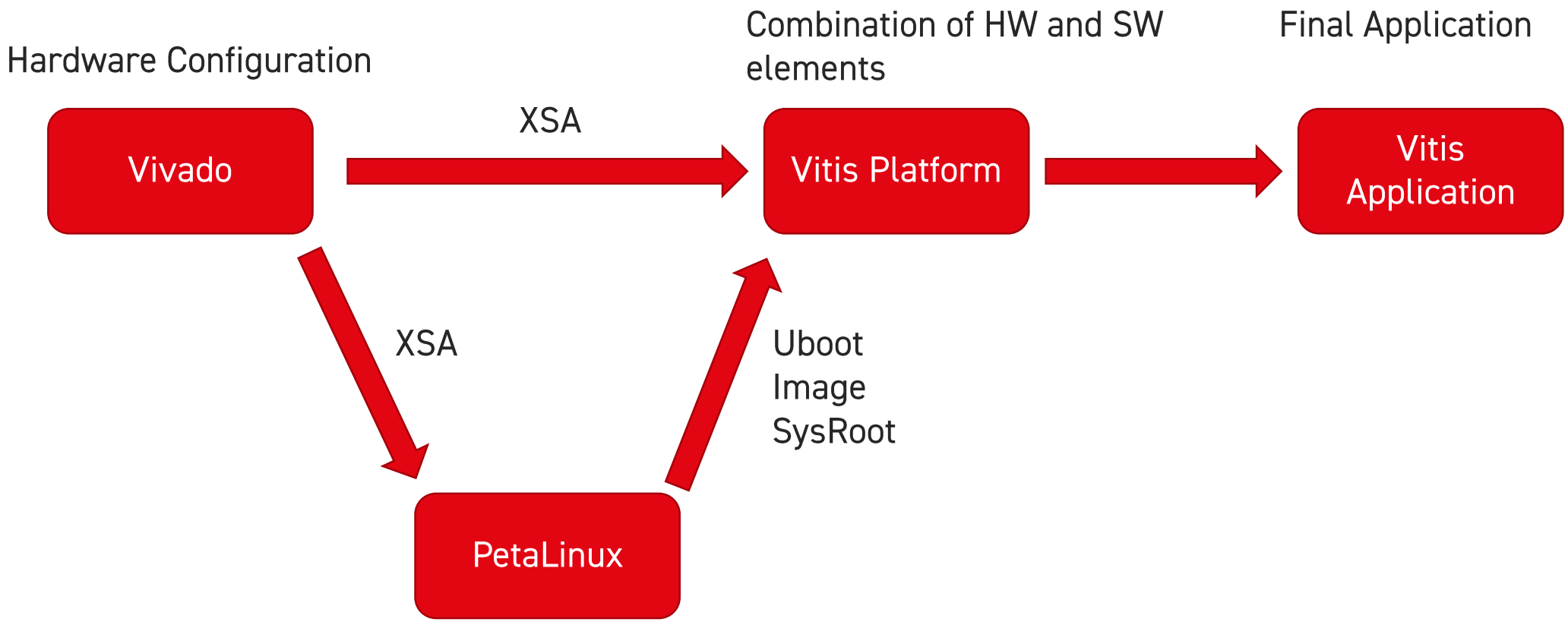
Embedded

- Traditional embedded development flow
  - Bare Metal
  - Free RTOS
  - PetaLinux
- Supports A9, A53, R5, PMU & Microblaze





Vitis Flow


Hardware Configuration
Combination of HW and SW elements
Final Application
Vivado
XSA
Vitis Platform
Vitis Application
XSA
Uboot
Image
SysRoot
PetaLinux

Vitis libraries

- Common libraries: These libraries provide basic functions which are used across a range of applications and domains for example math, DSP and linear algebra.

- Domain specific: These libraries provide acceleration functions for specific domains, e.g. security, vision, finance or database.

  - Level one: The lowest level of implementation, intended for use in a High-Level Synthesis flow.
  - Level two: Middle level provides acceleration kernels that are used in the Vitis design flow and the Xilinx RunTime (XRT).
  - Level three: Highest level provides applications created from several acceleration kernels. This uses API and of course XRT.

Vitis Libraries https://xilinx.github.io/Vitis_Librar...

146 commits | 2 branches | 0 releases | 1 contributor | Apache-2.0

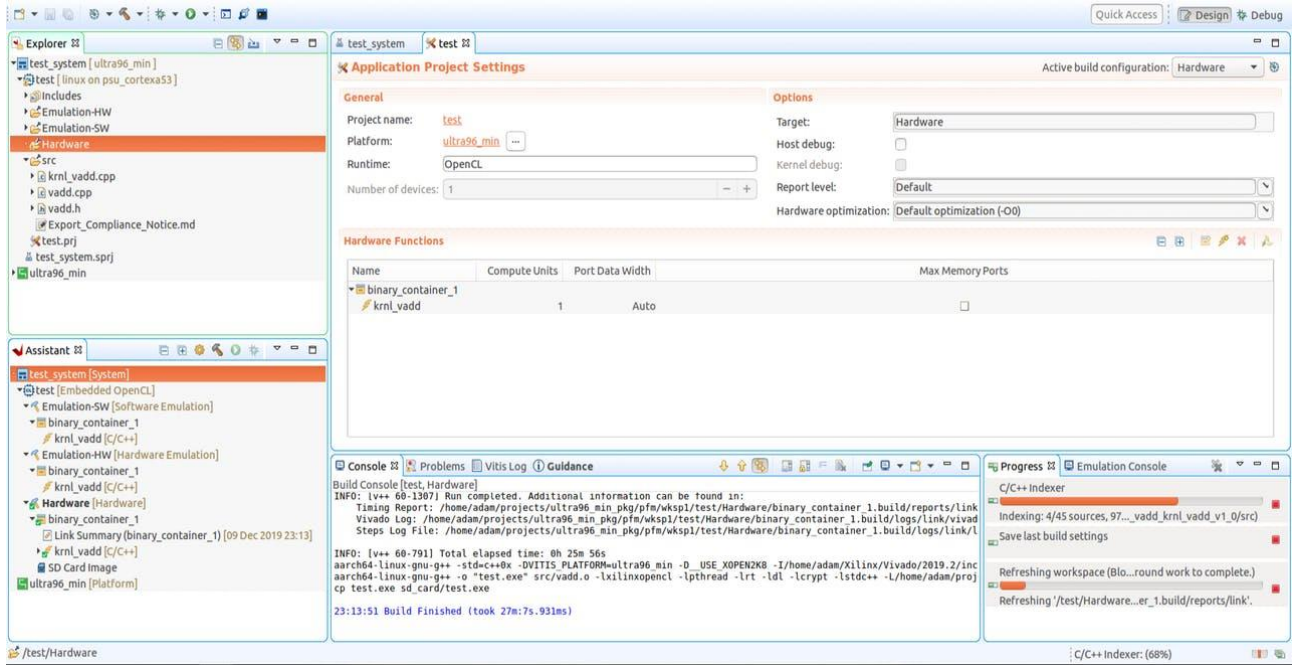
Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

luciferlee merge data_compression — Latest commit b658aa5 17 days ago

blas	merge blas	20 days ago
data_compression	merge data_compression	17 days ago
database	merge database	20 days ago
dsp	merge dsp	17 days ago
quantitative_finance	merge quantitative_finance	19 days ago
security	merge security	20 days ago
solver	Merge commit 'f3dd055984097e3df6b56ef6c22d2c2b7608b762'	2 months ago
utils	Merge commit 'ad4d3bd1c69bd1e920ba629efbeeba4cfe0d294d'	2 months ago
vision	merge vision	17 days ago
Jenkinsfile	add top level Jenkinsfile	19 days ago
LICENSE.txt	Create LICENSE.txt	2 months ago
README.md	update readme	last month

Build Flow

- Accelerated flow, building the design each time is time consuming
- Three different build flows are available
  - Software emulation: Runs both the host and the kernel on the x86.
  - Hardware emulation: Runs the kernel as a compiled hardware model, while the host uses a C simulator.
  - Hardware: Generates the final bootable image for the system



Build Flow

test_system | test | vadd.cpp | krnl_vadd.cpp | makefile

Application Project Settings

Active build configuration: Hardware
- Emulation-SW
- Emulation-HW
- Hardware

General

Project name: test
Platform: ultra96_min
Runtime: OpenCL
Number of devices: 1

Options

Target: Hardware
Host debug:
Kernel debug:
Report level: Default
Hardware optimization: Default optimization (-O0)

Hardware Functions

Name	Compute Units	Port Data Width	Max Memory Ports

microzed | microzed_tst_system | microzed_tst | vadd.cpp | vadd.h | vivado.log

- microzed
  - ps7_cortexa9
    - linux on ps7_cortexa9
      - Libraries
  - ps7_cortexa9_0
    - zynq_fsbl
      - Board Support Package

Domain: linux_domain

OS: linux
Processor: ps7_cortexa9
Supported Runtimes: OpenCL
Display Name: linux on ps7_cortexa9
Description: linux_domain

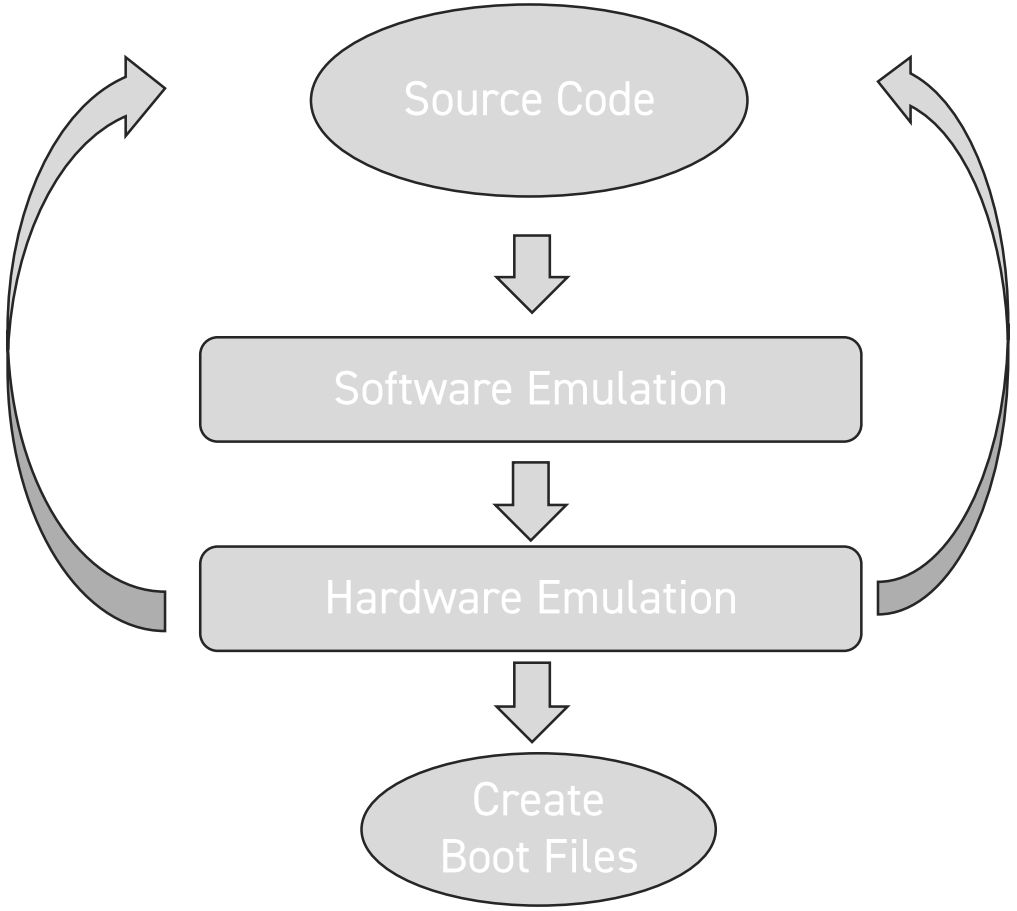
Bif File: /home/adam/projects/microzed_pkg/pfm/boot/linux.bif
Boot Components Directory: /home/adam/projects/microzed_pkg/pfm/boot
Linux Image Directory: /home/adam/projects/microzed_pkg/pfm/boot
Sysroot Directory: /home/adam/projects/microzed_pkg/pfm/sysroots/cortexa9t2hf-neon-xilinx-linux-gnue
QEMU Data: /home/adam/projects/microzed_pkg/pfm/boot/qemu
QEMU Arguments: /home/adam/projects/microzed_pkg/pfm/boot/qemu/qemu_args.txt

Flow Selection

For Emulation these need to be completed

Emulation Flow

Optimize Performance
Optimize Interfacing
Optimize Resources



Syntax Errors
Verify Functional
Performance

Creating a Platform

- First, we need a Linux Machine
- I use a Linux VM on a USB C SSD and works very well
- AWS Instances work brilliantly too

Component	Requirement
Operating System	Linux, 64-bit: • Ubuntu 16.04.5 LTS, 16.04.6 LTS, 18.04.1 LTS, 18.04.2 LTS • CentOS 7.4, 7.5, 7.6 • RHEL 7.4, 7.5, 7.6
System Memory	For Alveo cards: 64 GB (80 GB is recommended) For embedded: 32 GB
Internet Connection	Required for downloading drivers and utilities.
Hard disk space	100 GB

XRT Run Time

- Once Vitis is installed we need to install the XRT
- Answer Record 73055 very useful if issues

```
adam@adam-VirtualBox: ~/Downloads
First Installation: checking all kernels...
Building only for 4.15.0-70-generic
Building initial module for 4.15.0-70-generic
Done.

xocl:
Running module version sanity check.
 - Original module
   - No original module exists within this kernel
 - Installation
   - Installing to /lib/modules/4.15.0-70-generic/updates/dkms/

xclmgmt.ko:
Running module version sanity check.
 - Original module
   - No original module exists within this kernel
 - Installation
   - Installing to /lib/modules/4.15.0-70-generic/updates/dkms/

depmod....

DKMS: install completed.
Finished DKMS common.postinst
Loading new XRT Linux kernel modules
Installing MSD / MPD daemons
Skipping pyopencl installation...
adam@adam-VirtualBox:~/Downloads$
```

Installing Xilinx Runtime

Xilinx Runtime (XRT) is implemented as a combination of user-space and kernel driver components. XRT supports Alveo PCIe-based cards, as well as Zynq UltraScale+ MPSoC-based embedded system platforms, and provides a software interface to Xilinx programmable logic devices.

You only need to install XRT once, regardless of how many platforms you may be installing.

⚠ **IMPORTANT:** XRT installation uses standard Linux RPM and Linux DEB distribution files, and root access is required for all software and firmware installations.

`<rpm-dir>` or `<deb-dir>` is the directory where you downloaded the packages to install.

To download and install the XRT package for your operating system, do the following.

CentOS/RedHat

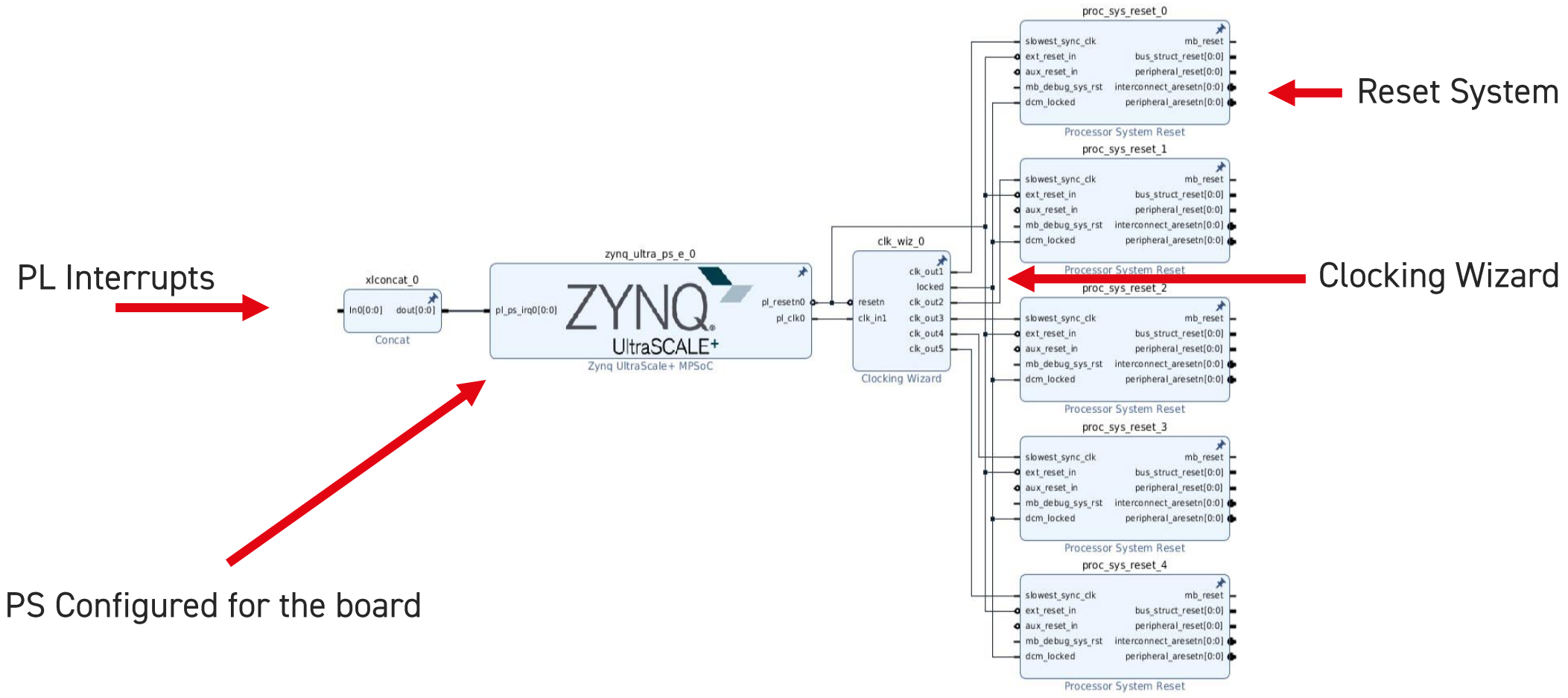
1. To download the RPM file, click this link.
2. To install the package, enter the following command.

```
sudo yum install <rpm-dir>/<xrt_filename>.rpm
```

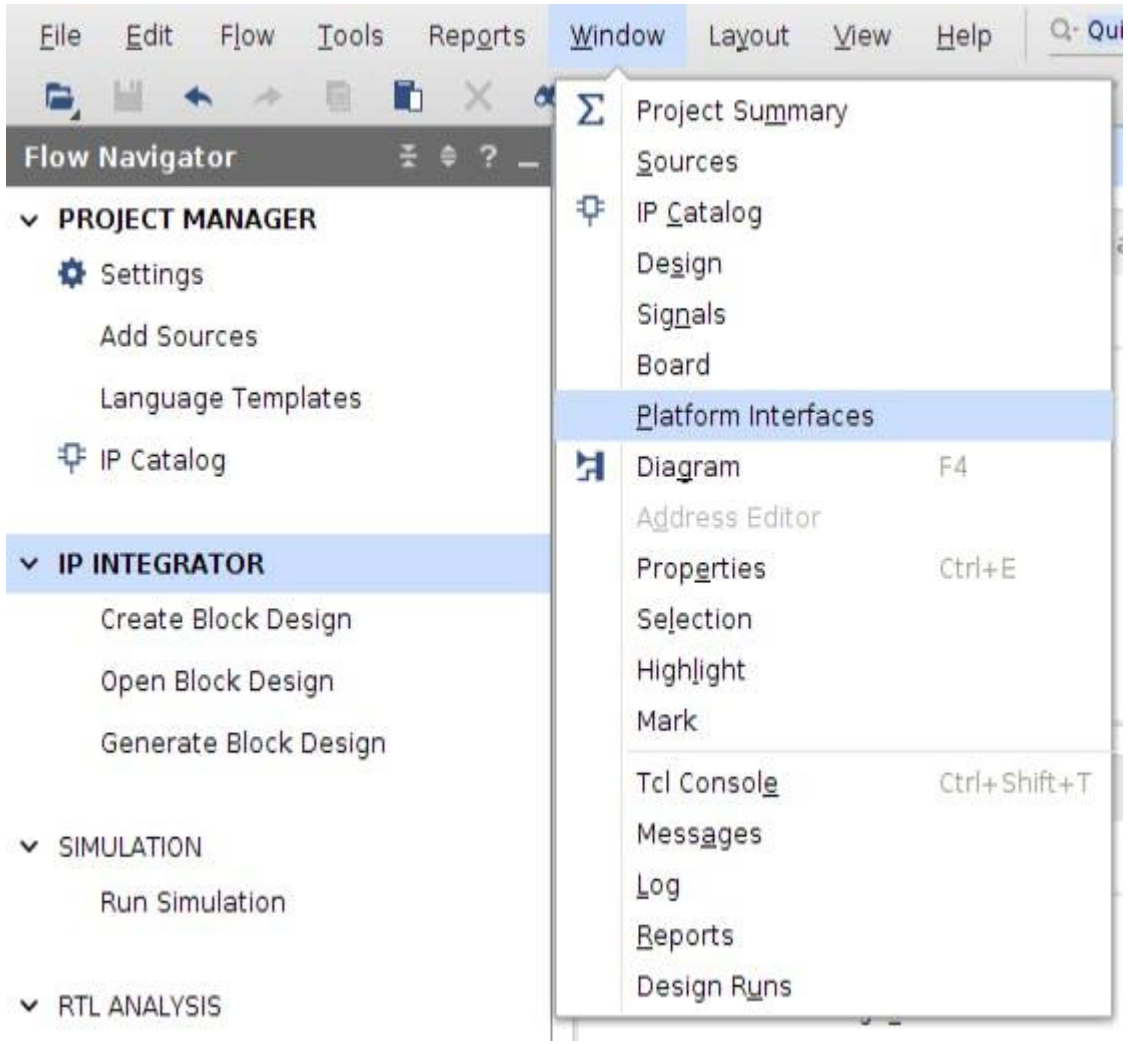
Ubuntu

1. To download the DEB file, click one of the following:
   - Ubuntu 16.04

Vivado Design

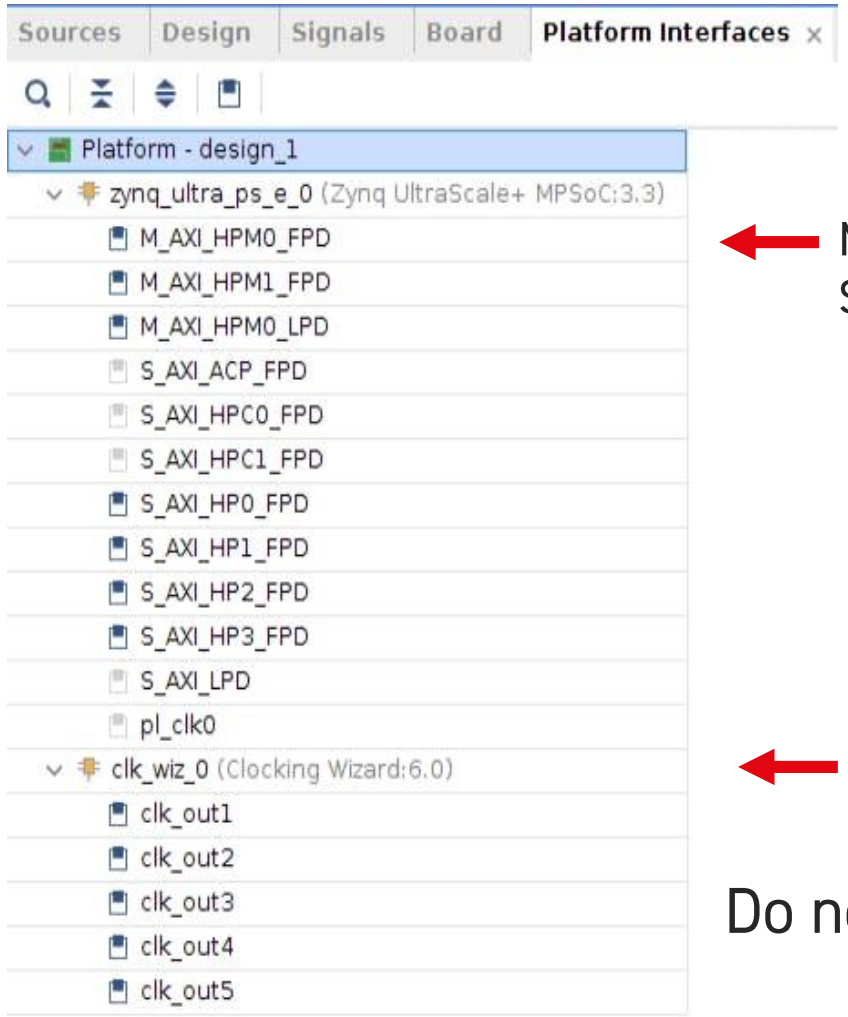


Vivado Design



Declare capabilities which may or may not be made available to the V++ compiler.

Vivado Platform Interfaces



M_AXI FPD are used for configuration
S_AXI used for data transfer

Enable the different clocks

Do not forget to enable the interrupts, not shown.

Vivado Generating the XSA

```
set_property platform.design_intent.embedded true [current_project]
set_property platform.design_intent.server_managed false [current_project]
set_property platform.design_intent.external_host false [current_project]
set_property platform.design_intent.datacenter false [current_project]
set_property platform.default_output_type "sd_card" [current_project]
```

Add Information to our Platform

```
write_hw_platform -include_bit ultra96_min.xsa
validate_hw_platform ./ultra96_min.xsa
```

Write out and Validate the XSA

Tcl Console | Messages | Log | Reports | Design Runs

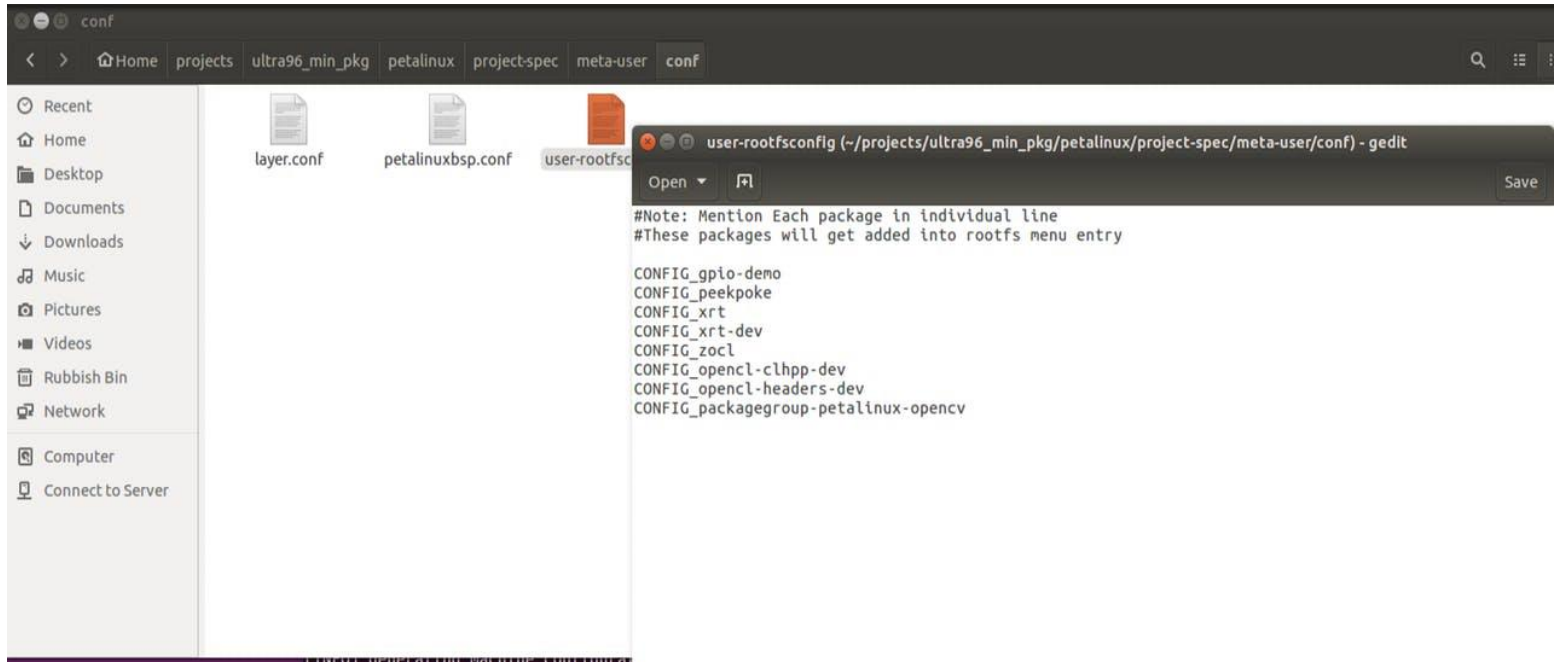
```
INFO: [Vivado 12-4896] Successfully created Hardware Platform: /home/adam/projects/ultra96_min_pkg/vivado/ultra96_min.xsa
write_hw_platform: Time (s): cpu = 00:00:10 ; elapsed = 00:00:10 . Memory (MB): peak = 8615.188 ; gain = 0.000 ; free physical = 10089 ; free virtual = 13700
/home/adam/projects/ultra96_min_pkg/vivado/ultra96_min.xsa
validate_hw_platform ./ultra96_min.xsa
INFO: [Vivado 12-6074] Validating Hardware Platform: './ultra96_min.xsa'
INFO: [Vivado 12-8115] Found XML metadata file: xsa.xml
INFO: [Vivado 12-6078] Validating platform properties...
INFO: [Vivado 12-6079] Validating unified platform...
INFO: [Vivado 12-6073] Validating 'pre_synth' platform state...
INFO: [Vivado 12-6077] Validating platform files...
INFO: [Vivado 12-6067] Found file 'emu/emu.xml' of type 'EMU_XML' in the Hardware Platform.
INFO: [Vivado 12-6067] Found file 'ultra96_min.bit' of type 'FULL_BIT' in the Hardware Platform.
INFO: [Vivado 12-6067] Found file 'ultra96_min.hpfm' of type 'HPFM' in the Hardware Platform.
INFO: [Vivado 12-6067] Found file 'prj/rebuild.tcl' of type 'REBUILD_TCL' in the Hardware Platform.
INFO: [Vivado 12-6066] Finished running validate_dsa for file: './ultra96_min.xsa'
```

Validated XSA which can be exported

PetaLinux

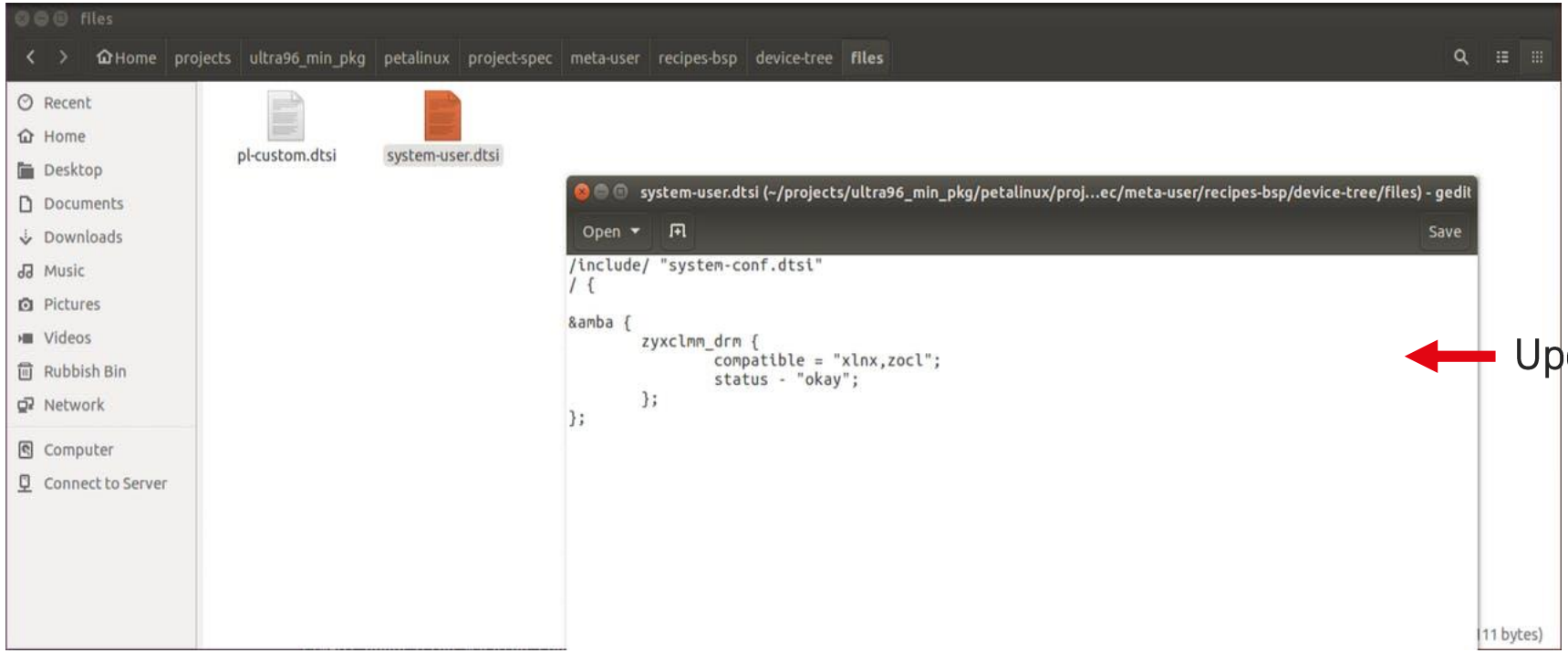
Create a new PetaLinux project based off the XSA

Need to make several modifications to the OS, Device Tree and RootFS to support XRT



Update RootFS packages

PetaLinux



Update the device tree

PetaLinux

petalinux-config -c rootfs

```
adam@adam-VirtualBox: ~/projects/ultra96_min_pkg/petalinux
/home/adam/projects/ultra96_min_pkg/petalinux/project-spec/configs/rootfs_config - Configuration
→ user packages
                                  user packages
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted
  letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press
  <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module
  < > module capable

                [*] opencl-clhpp-dev
                [*] opencl-headers-dev
                [*] packagegroup-petalinux-opencv
                [*] xrt
                [*] xrt-dev
                [*] zocl

                <Select>    < Exit >    < Help >    < Save >    < Load >
```

Notice new RootFS user packages

PetaLinux

petalinux-config -c kernel

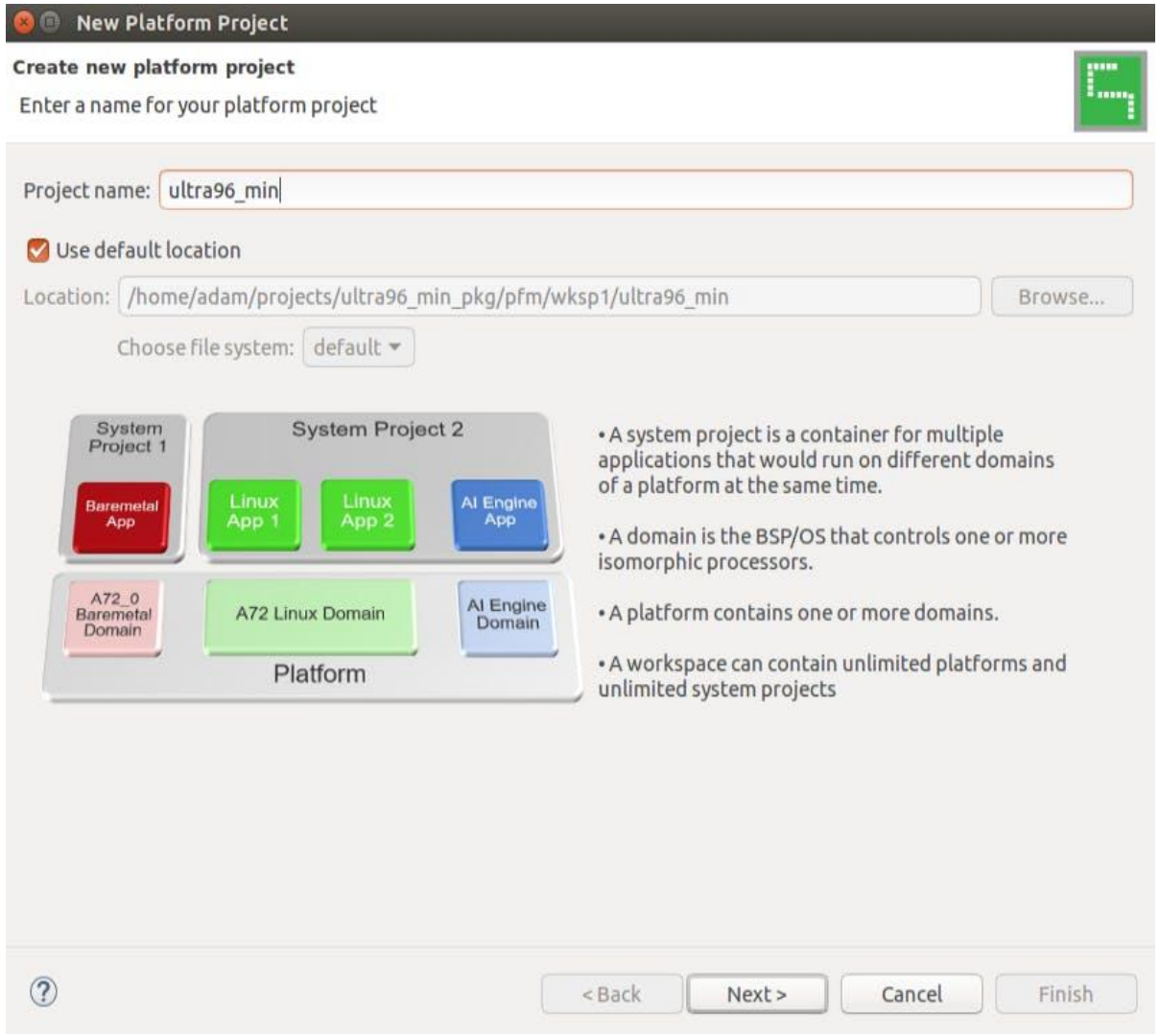
- Device Drivers > Generic Driver Options > DMA Contiguous Memory Allocator > Size in Mega Bytes change the size from 256 to 1024 MB
- Device Drivers -> Staging drivers -> Xilinx APF Accelerator driver
- Device Drivers -> Staging drivers -> Xilinx APF Accelerator driver -> Xilinx APF DMA engines support

petalinux-build ← Creates OS and supporting files

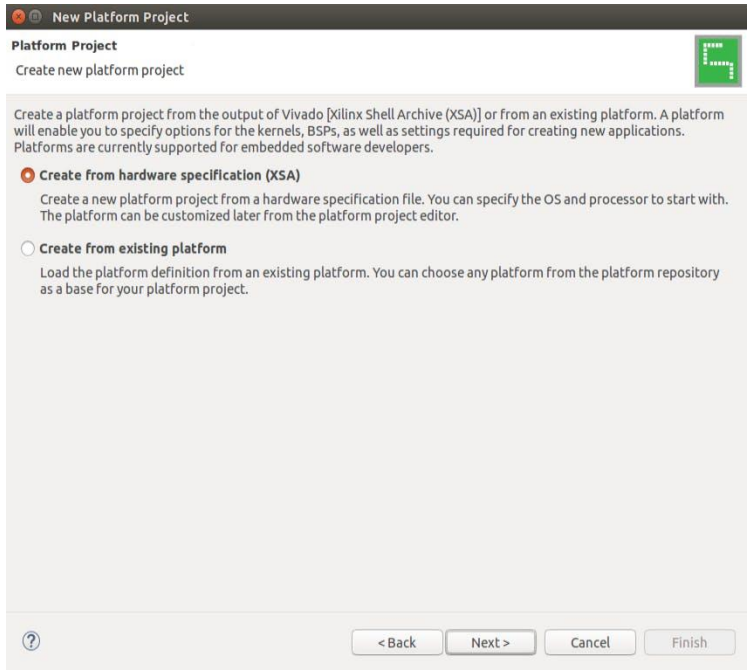
petalinux-build --sdk ← Create Sys Root needed for Vitis Compilation

./sdk.sh ← Install SysRoot

Vitis



Create a new Platform



Vitis

New Platform Project

Platform Project Specification

Provide the hardware and software specification for the new platform project

Hardware Specification

XSA file: /home/adam/projects/ultra96_min_pkg/vivado/ultra96_min.xsa Browse...

Software Specification

Operating system: linux

Processor: psu_cortexa53

Note: The Linux domain added to the platform project needs more
specify the missing details in the platform project editor before ger

Generate boot components

< Back

Use the XSA from Vivado

ultra96_min

type filter text

- ultra96_min
  - psu_cortexa53
    - linux on psu_cortexa53
      - Libraries
  - psu_cortexa53_0
    - zynqmp_fsbl
      - Board Support Package
  - psu_pmu_0
    - zynqmp_pmufw
      - Board Support Package

Domain: linux_domain

OS:	linux
Processor:	psu_cortexa53
Supported Runtimes:	OpenCL
Display Name:	linux on psu_cortexa53
Description:	linux_domain
Bif File:	/home/adam/projects/ultra96_min_pkg/pfm/boot/linux.bif Browse...
Boot Components Directory:	/home/adam/projects/ultra96_min_pkg/pfm/boot Browse...
Linux Image Directory:	/home/adam/projects/ultra96_min_pkg/pfm/boot Browse...
Sysroot Directory:	/home/adam/projects/ultra96_min_pkg/pfm/sysroots/aarch64-xilinx-linux Browse...
QEMU Data:	Browse...
QEMU Arguments:	Browse...
PMU QEMU Arguments:	Browse...

Main Hardware Specification

Provided by PetaLinux Stage

Vitis

New Platform

Explorer
- ultra96_min
- bitstream
- export
- hw
- logs
- resources
- zynqmp_fsbl
- zynqmp_pmufw
- platform.spr

Assistant
- ultra96_min [Platform]

Domain: linux_domain

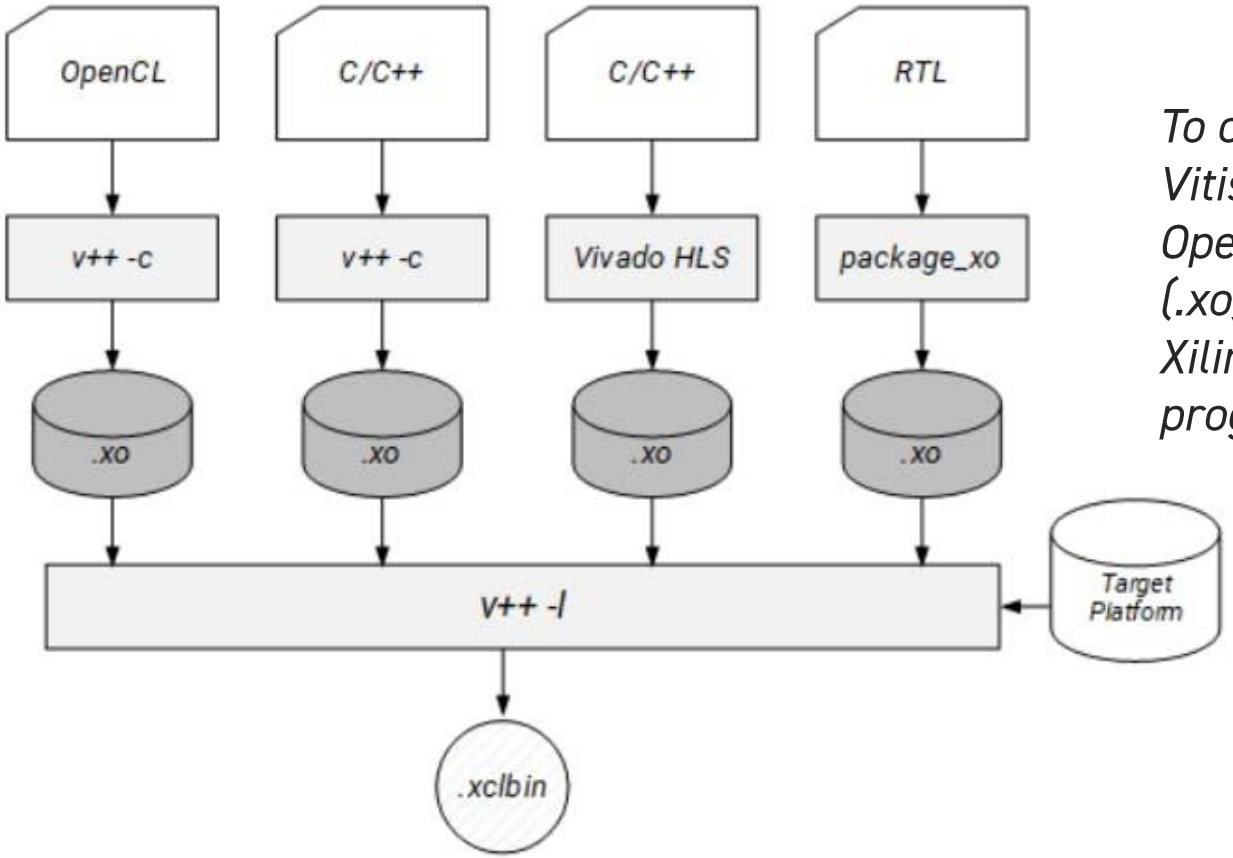
OS:	linux
Processor:	psu_cortexa53
Supported Runtimes:	OpenCL
Display Name:	linux on psu_cortexa53
Description:	linux_domain
Bif File:	/home/adam/projects/ultra96_min_pkg/pfm/boot/linux.bif
Boot Components Directory:	/home/adam/projects/ultra96_min_pkg/pfm/boot
Linux Image Directory:	/home/adam/projects/ultra96_min_pkg/pfm/boot
Sysroot Directory:	/home/adam/projects/ultra96_min_pkg/pfm/sysroots/aarch64-xilinx-linux
QEMU Data:	
QEMU Arguments:	
PMU QEMU Arguments:	

Console

```
Build Console [ultra96_min]
ptw_mod_dap.o  xpfw_platform.o  pm_extern.o  pm_node.o  xpfw_core.o  pm_periph.o  xpfw_mod_common.o  pm_pll.o  pm_gpp.o  pm_usb
.o  pm_power.o  xpfw_main.o  pm_reset.o  xpfw_mod_sched.o  pm_csudma.o  pm_clock.o  xpfw_start.o  -MMD -MP      -mlittle-endian
 -mxl-barrel-shift -mxl-pattern-compare -mcpu=v9.2 -mxl-soft-mul -Os -flto -ffat-lto-objects    -Wl,--start-group,-lxil,-lgcc,-
lc,--end-group -Wl,--start-group,-lxilfpga,-lxil,-lxilsecure,-lgcc,-lc,--end-group -Wl,--start-group,-lxilsecure,-lxil,-lgcc,-l
c,--end-group -Wl,--start-group,-lxilskey,-lxil,-lgcc,-lc,--end-group
                                        -nostartfiles -Wl,--no-relax -Wl,--gc-sections -Lzynqmp_pmufw_bsp/psu_pmu_0/li
b -Tlscript.ld

Copying the sysroot data, this may take few minutes...

22:41:17 Build Finished (took 1m:13s.826ms)
```

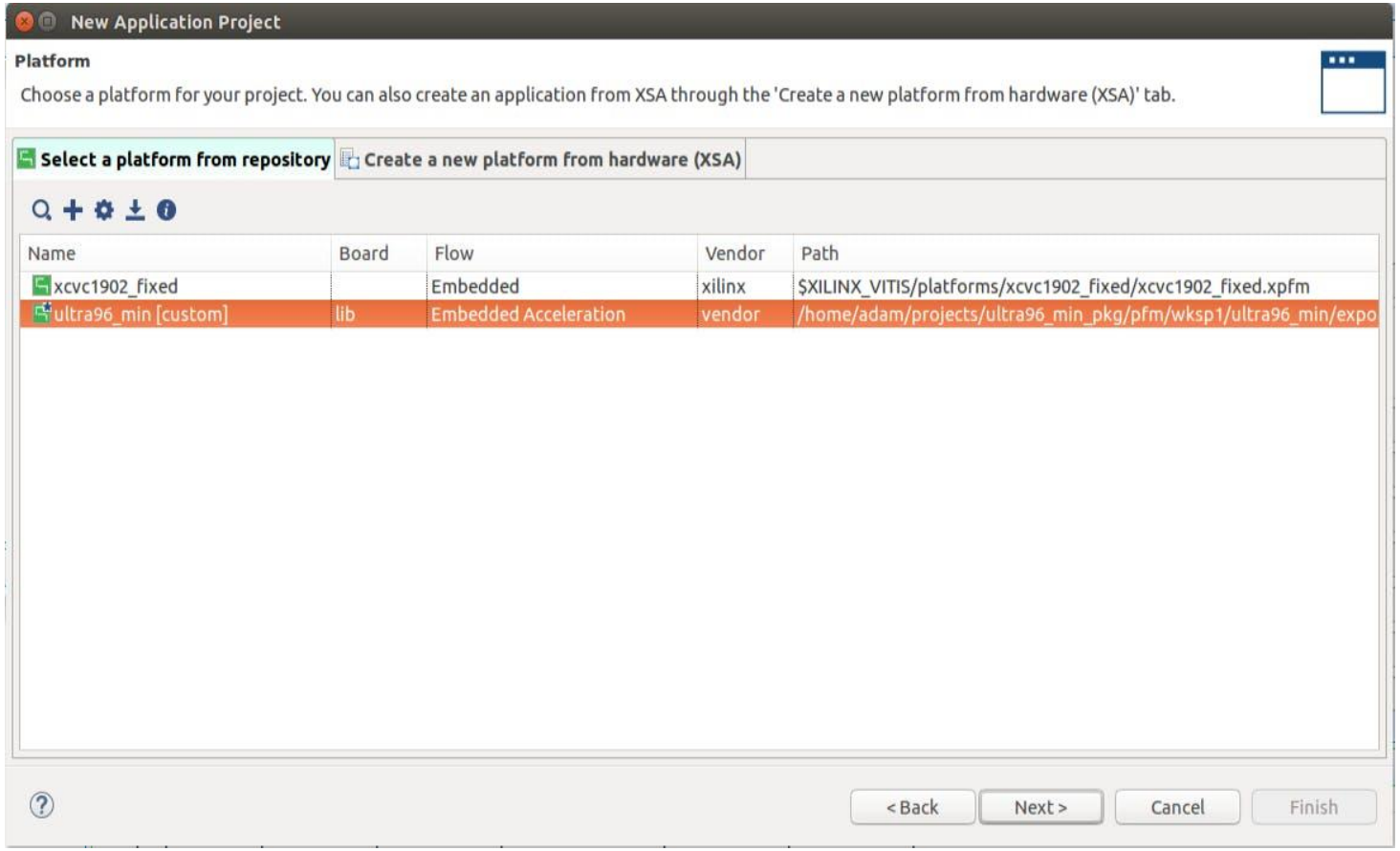
Platform build status

Vitis



To create programmable logic kernels, the Vitis V++ Compiler is capable of compiling OpenCL, C/C++ and RTL into Xilinx Object (.xo) file and then linking these into the Xilinx binary (.xclbin) for the programmable logic.

Vitis Project



Newly Created Platform

Vitis Project

Application

Kernel

Host

Accelerated Kernel

Explorer
- test_system [ultra96_min]
  - test [linux on psu_cortexa53]
    - Includes
    - Emulation-HW
    - Emulation-SW
    - Hardware
    - src
      - krnl_vadd.cpp
      - vadd.cpp
      - vadd.h
      - Export_Compliance_Notice.md
    - test.prj
  - test_system.sprj
- ultra96_min

Assistant
- test_system [System]
- test [Embedded OpenCL]
  - Emulation-SW [Software Emulation]
    - binary_container_1
      - krnl_vadd [C/C++]
  - Emulation-HW [Hardware Emulation]
    - binary_container_1
      - krnl_vadd [C/C++]
  - Hardware [Hardware]
    - binary_container_1
      - Link Summary (binary_container_1) [09 Dec 2019 23:13]
      - krnl_vadd [C/C++]
    - SD Card Image
- ultra96_min [Platform]

Application Project Settings — Active build configuration: Hardware

General
- Project name: test
- Platform: ultra96_min
- Runtime: OpenCL
- Number of devices: 1

Options
- Target: Hardware
- Host debug:
- Kernel debug:
- Report level: Default
- Hardware optimization: Default optimization (-O0)

Hardware Functions

Name	Compute Units	Port Data Width	Max Memory Ports
binary_container_1			
krnl_vadd	1	Auto	

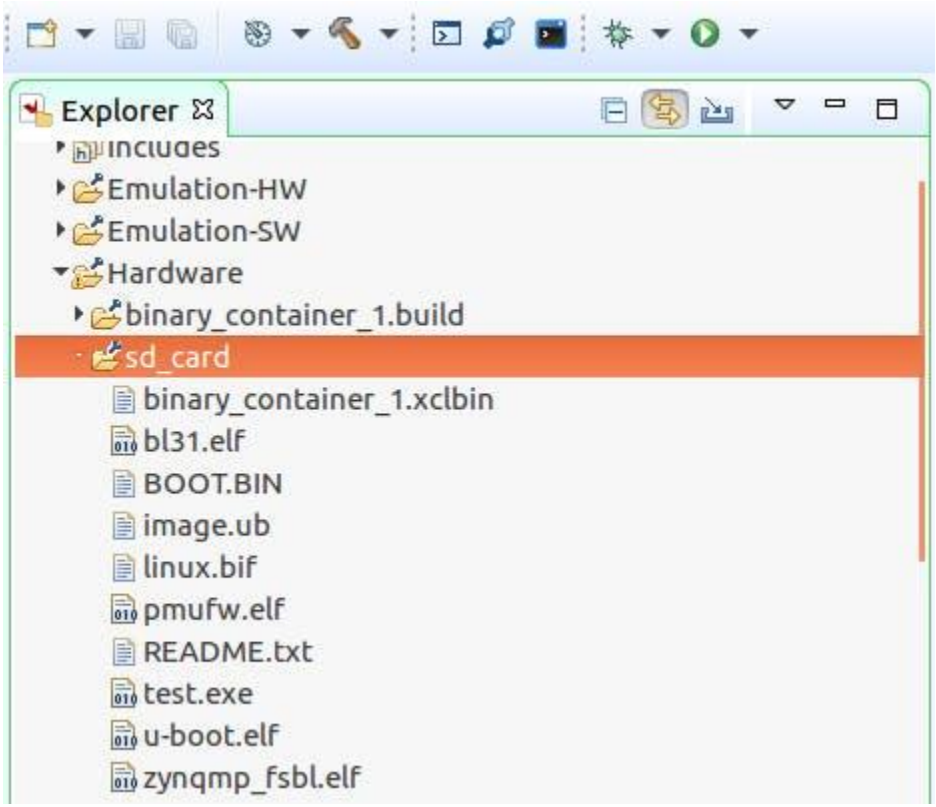
Console — Build Console [test, Hardware]

```
INFO: [v++ 60-1307] Run completed. Additional information can be found in:
    Timing Report: /home/adam/projects/ultra96_min_pkg/pfm/wksp1/test/Hardware/binary_container_1.build/reports/link
    Vivado Log: /home/adam/projects/ultra96_min_pkg/pfm/wksp1/test/Hardware/binary_container_1.build/logs/link/vivad
    Steps Log File: /home/adam/projects/ultra96_min_pkg/pfm/wksp1/test/Hardware/binary_container_1.build/logs/link/l

INFO: [v++ 60-791] Total elapsed time: 0h 25m 56s
aarch64-linux-gnu-g++ -std=c++0x -DVITIS_PLATFORM=ultra96_min -D__USE_XOPEN2K8 -I/home/adam/Xilinx/Vivado/2019.2/inc
aarch64-linux-gnu-g++ -o "test.exe" src/vadd.o -lxilinxopencl -lpthread -lrt -ldl -lcrypt -lstdc++ -L/home/adam/proj
cp test.exe sd_card/test.exe

23:13:51 Build Finished (took 27m:7s.931ms)
```

Progress
- C/C++ Indexer — Indexing: 4/45 sources, 97..._vadd_krnl_vadd_v1_0/src)
- Save last build settings
- Refreshing workspace (Blo...round work to complete.) — Refreshing '/test/Hardware...er_1.build/reports/link'.

/test/Hardware — C/C++ Indexer: (68%)

Vitis Project



export XILINX_XRT=/usr

./test.exe binary_container_1.xclbin

Vitis Project

COM4 - PuTTY

```
Starting internet superserver: inetd.
Starting syslogd/klogd: done
Starting tcf-agent: OK

PetaLinux 2019.2 ultra96_min /dev/ttyPS0

ultra96_min login: root
Password:
root@ultra96_min:~#
root@ultra96_min:~# cd ../..
root@ultra96_min:/# cd run/media
root@ultra96_min:/run/media# ls
mmcblk0p1  mmcblk0p2
root@ultra96_min:/run/media# cd mmcblk0p1
root@ultra96_min:/run/media/mmcblk0p1# export XILINX_XRT=/usr
root@ultra96_min:/run/media/mmcblk0p1# ./test.exe binary_container_1.xclbin
[  132.942123] [drm] Pid 2265 opened device
[  132.946095] [drm] Pid 2265 closed device
[  132.950426] [drm] Pid 2265 opened device
Loading: 'binary_container_1.xclbin'
[  133.221515] [drm] Finding IP_LAYOUT section header
[  133.221527] [drm] Section IP_LAYOUT details:
[  133.226409] [drm]   offset = 0x54fca0
[  133.230674] [drm]   size = 0x58
[  133.234341] [drm] Finding DEBUG_IP_LAYOUT section header
[  133.237471] [drm] AXLF section DEBUG_IP_LAYOUT header not found
[  133.242784] [drm] Finding CONNECTIVITY section header
[  133.248701] [drm] Section CONNECTIVITY details:
[  133.253754] [drm]   offset = 0x54fcf8
[  133.258282] [drm]   size = 0x28
[  133.261942] [drm] Finding MEM_TOPOLOGY section header
[  133.265073] [drm] Section MEM_TOPOLOGY details:
[  133.270124] [drm]   offset = 0x54fbf8
[  133.274653] [drm]   size = 0xa8
[  133.280177] [drm] No ERT scheduler on MPSoC, using KDS
[  133.288895] [drm] Fail to install CU 0 interrupt handler: -22. Fall back to polling mode.
[  133.297073] [drm] scheduler config ert(0)
[  133.297075] [drm]   cus(1)
[  133.301084] [drm]   slots(16)
[  133.303784] [drm]   num_cu_masks(1)
[  133.306742] [drm]   cu_shift(16)
[  133.310224] [drm]   cu_base(0xa0000000)
TEST PASSED
[  133.313438] [drm]   polling(1)
[  133.327924] [drm] Pid 2265 closed device
root@ultra96_min:/run/media/mmcblk0p1#
```

XRT and Execution

Kernel loading

Test Run

Vitis Example

```
cl::Platform::get(&platforms);
for(size_t i = 0; (i < platforms.size() ) & (found_device == false) ;i++){
    cl::Platform platform = platforms[i];
    std::string platformName = platform.getInfo<CL_PLATFORM_NAME>();
    if ( platformName == "Xilinx"){
        devices.clear();
        platform.getDevices(CL_DEVICE_TYPE_ACCELERATOR, &devices);
    if (devices.size()){
        device = devices[0];
        found_device = true;
        break;
    }
    }
}
```

Find Available Platforms

```
// Creating Context and Command Queue for selected device
cl::Context context(device);
cl::CommandQueue q(context, device, CL_QUEUE_PROFILING_ENABLE);
// Load xclbin
std::cout << "Loading: '" << xclbinFilename << "'\n";
std::ifstream bin_file(xclbinFilename, std::ifstream::binary);
bin_file.seekg (0, bin_file.end);
unsigned nb = bin_file.tellg();
bin_file.seekg (0, bin_file.beg);
char *buf = new char [nb];
bin_file.read(buf, nb);
// Creating Program from Binary File
cl::Program::Binaries bins;
bins.push_back({buf,nb});
devices.resize(1);
cl::Program program(context, devices, bins);
```

Create Context
Program

Vitis Project

```
cl::Buffer buffer_a(context, CL_MEM_READ_ONLY, size_in_bytes);
cl::Buffer buffer_b(context, CL_MEM_READ_ONLY, size_in_bytes);
cl::Buffer buffer_result(context, CL_MEM_WRITE_ONLY, size_in_bytes);
```

Allocate Buffers

```
int *ptr_a = (int *) q.enqueueMapBuffer (buffer_a , CL_TRUE , CL_MAP_WRITE ,
0, size_in_bytes);
int *ptr_b = (int *) q.enqueueMapBuffer (buffer_b , CL_TRUE , CL_MAP_WRITE ,
0, size_in_bytes);
int *ptr_result = (int *) q.enqueueMapBuffer (buffer_result , CL_TRUE , CL_MA
P_READ , 0, size_in_bytes);
```

Map buffers to the kernel

```
// Data will be migrated to kernel space
q.enqueueMigrateMemObjects({buffer_a,buffer_b},0/* 0 means from host*/);
//Launch the Kernel
q.enqueueTask(krnl_vector_add);
// The result of the previous kernel execution will need to be retrieved in
// order to view the results. This call will transfer the data from FPGA to
// source_results vector
q.enqueueMigrateMemObjects({buffer_result},CL_MIGRATE_MEM_OBJECT_HOST);
q.finish();
```

Migrate and execute kernel

Vitis Example

```
void krnl_vadd(const unsigned int *in1, // Read-Only Vector 1
const unsigned int *in2, // Read-Only Vector 2
unsigned int *out_r,     // Output Result
int size                 // Size in integer
) {
#pragma HLS INTERFACE m_axi port = in1 offset = slave bundle = gmem
#pragma HLS INTERFACE m_axi port = in2 offset = slave bundle = gmem
#pragma HLS INTERFACE m_axi port = out_r offset = slave bundle = gmem
#pragma HLS INTERFACE s_axilite port = in1 bundle = control
#pragma HLS INTERFACE s_axilite port = in2 bundle = control
#pragma HLS INTERFACE s_axilite port = out_r bundle = control
#pragma HLS INTERFACE s_axilite port = size bundle = control
#pragma HLS INTERFACE s_axilite port = return bundle = control
```

Interfaces

Interface Declarations

Vitis Project

```
for (int i = 0; i < size; i += BUFFER_SIZE) {
    #pragma HLS LOOP_TRIPCOUNT min=c_len max=c_len
    int chunk_size = BUFFER_SIZE;
    //boundary checks
    if ((i + BUFFER_SIZE) > size)
        chunk_size = size - i;
    read1: for (int j = 0; j < chunk_size; j++) {
        #pragma HLS LOOP_TRIPCOUNT min=c_size max=c_size
        #pragma HLS PIPELINE II=1
        v1_buffer[j] = in1[i + j];
    }
    //Burst reading B and calculating C and Burst writing
    // to  Global memory
    vadd_writeC: for (int j = 0; j < chunk_size; j++) {
        #pragma HLS LOOP_TRIPCOUNT min=c_size max=c_size
        #pragma HLS PIPELINE II=1
        //perform vector addition
        out_r[i+j] = v1_buffer[j] + in2[i+j];
    }
}
```

Pragma Trip Count = enables HLS tool to understand the number of loop iterations

Pragma Pipeline Initiation Interval = the target number of clocks between being able to process new inputs

Questions?

PLC2

PROGRAMMABLE LOGIC
COMPETENCE CENTER

PLC2 GmbH
Hugstmattweg 30
D-79112 Freiburg
Germany

Fon +49 (0) 7664-913 13-0
Fax +49 (0) 7664-913 13-99
E-Mail: info@plc2.de
Internet: www.plc2.com