# Table of Contents

# YUGE at Northwestern University

This page documents procedures for setting up firmware and software on the YUGE board at Northwestern University. While some of the steps listed below will be specific to our set up, we hope this provides some useful guidance for others pursuing similar tasks.

# uHAL on ZYNQ

Our uHAL set up is largely based on the procedures described at http://gauss.bu.edu/svn/common-atca-blade.software/ipbus/trunk/ipbus-zynq-2019-05-03/ (E. Hazen et al. at BU)

## Image of the Petalinux file system on the YUGE

This is needed for the cross-compilation of the IPbus software suite. The image is created from the `rootfs.tgz` file.

```
cp /root/YUGEboot/for_kh/rootfs.tgz ./ mkdir rootfs tar -xf rootfs.tgz -C rootfs/ dd if=/dev/zero of=rootfs.ext4  bs=4k count=60000 mkfs.ext4 rootfs.ext4 tune2fs -c0 -i0 rootfs.ext4
```

## Cross-compile Boost C++ library

Boost 1.64.0 is cross-compiled for the YUGE Petalinux. (details?)

## Check out the IPbus software package

For cross-compilation, we need to checkout a specific commit of the IPbus software repository, and then apply a patch (`ipbus.diff`) that can be found here.

```
git clone https://github.com/ipbus/ipbus-software.git cd ipbus-software/ git checkout 24db2eb4c1bffe587011eb662bbf2ad9e231657f git apply < ../ipbus.diff
```

## Building the IPbus software suite

Mount the Petalinux file system image.

```
mount -o loop rootfs.ext4 /mnt/zynq_root ZYNQROOT=/mnt/zynq_root
```

Several environment variables are declared for later convenience.

```
BOOSTDIR=/scratch/boost_cxx export PATH=/scratch/python2.7/bin:${PATH} export PYTHONPATH=/scratch/python2.7/lib export PATH=/scratch/tar_1_24/bin:${PATH} export PATH=/scratch/socat_2_0_0/bin:${PATH} PLNXDIR=/scratch/petalinux-v2017.2 source ${PLNXDIR}/settings.sh
```

A couple library files need to be added to the Petalinux filesystem. Also, a few soft-links need to be defined.

```
cp ${PLNXDIR}/tools/linux-i386/gcc-arm-linux-gnueabi/arm-linux-gnueabihf/libc/usr/lib/crti.o ${ZYNQROOT}/lib/ cp ${PLNXDIR}/tools/linux-i386/gcc-arm-linux-gnueabi/arm-linux-gnueabihf/libc/usr/lib/crtn.o ${ZYNQROOT}/lib/ cd ${ZYNQROOT}/lib ln -s libpthread-2.23.so ./libpthread.so ln -s
```

```
libm-2.23.so ./libm.so ln -s libc-2.23.so ./libc.so cd -
```

Compile.

```
cd ipbus-software make -j8 CXX=arm-linux-gnueabihf-c++ LD=arm-linux-gnueabihf-c++
LDFLAGS="--sysroot=${ZYNQROOT} -L${ZYNQROOT}/lib -L${BOOSTDIR}/lib" CXXFLAGS="-isystem
${ZYNQROOT} -I${BOOSTDIR}/include/  -g -O2 -std=c++11 -Wall -fPIC
-DDISABLE_PACKET_COUNTER_HACK" Set=uhal BUILD_PUGIXML=1 UHAL_NO_TESTS=1 UHAL_NO_PYTHON=1
```

Copy the uHAL libraries to the ZYNQ on the YUGE.

```
scp extern/pugixml/RPMBUILD/SOURCES/lib/libpugixml.so uhal/*/lib/* yuge:/usr/lib/
```

## uHAL example

Download the code from here. Before cross-compiling, a couple more libraries need to be copied to the Petalinux file system. The main source code for the example is `ipbus_test/src/common/test.cxx`. The example reads in the value at a register, performs a write to that register the given input at run time, and reads out the register again.

```
cp
${PLNXDIR}/tools/linux-i386/gcc-arm-linux-gnueabi/arm-linux-gnueabihf/libc/usr/lib/crt1.o
${ZYNQROOT}/lib/ cp
${PLNXDIR}/tools/linux-i386/gcc-arm-linux-gnueabi/arm-linux-gnueabihf/libc/usr/lib/libc_nonshared
${ZYNQROOT}/usr/lib/ cd ../ipbus_test make CXX=arm-linux-gnueabihf-c++
LDFLAGS="--sysroot=${ZYNQROOT} -L${BOOSTDIR}/lib" CXXFLAGS="-I${BOOSTDIR}/include/"
```

Copy the address mapping XML files and the executable to the ZYNQ on the YUGE.

```
scp *.xml build/bin/ipbus_test yuge:
```

- If needed, set up the Xilinx Virtual Cable on the YUGE

```
/etc/init.d/xvcServer-init stop cat system_top_6089_101_revA_jtag.bit > /dev/xdevcfg
/etc/init.d/xvcServer-init start
```

- If needed, load IPbus firmware on the KU 115. The bit file can be found here. Open the Hardware Manager in Vivado 2018.2 and program the KU 115. Note the IP address of the KU 115 is 192.168.200.16.

```
source /home/xilinx/Vivado/2018.2/settings64.sh vivado
```

Log in to the ZYNQ via minicom. Set the ZYNQ to be on the 192.168.200.* subnet and run the ipbus_test executable.

```
minicom ifconfig eth0 192.168.200.7 ./ipbus_test 0xdeadbeef
```

- Restore the IP address of the ZYNQ by simply disabling and enabling the eth0 interface

```
ifdown eth0 ifup eth0
```

# Chip-to-chip example with EMP framework

## Build firmware using IPbus Builder (IPBB)

Set up some environment variables.

```
export PATH=/scratch/python2.7/bin:${PATH} export PYTHONPATH=/scratch/python2.7/lib export LD_LIBRARY_PATH=/scratch/boost_1_53_0/lib:${LD_LIBRARY_PATH} export LD_LIBRARY_PATH=/opt/cactus/lib:${LD_LIBRARY_PATH} source ~xilinx/Vivado/2018.2/settings64.sh
```

Create an IPBB work area `p2fwk-work`:

```
ipbb init p2fwk-work cd p2fwk-work
```

Checkout the EMP framework (our working version is a particular commit):

```
ipbb add git ssh://git@gitlab.cern.ch:7999/p2-xware/firmware/emp-fwk.git cd src/emp-fwk git checkout 91f277b392d4387e4fd2a54d9ef7662ea9bf0b91 cd -
```

Checkout the MP7 repo:

```
ipbb add git ssh://git@gitlab.cern.ch:7999/cms-cactus/firmware/mp7.git -b ephemeral/phase2-vC
```

Checkout the IPbus firmware repo (our working version is a particular commit)

```
ipbb add git https://github.com/ipbus/ipbus-firmware cd src/ipbus-firmware git checkout 990d28da4a0a8e94f3b76d74069545d8f50efce9 cd -
```

Download a zip file containing YUGE board-specific files. The directory hierarchy of the files assumes you are in the IPBB work area.

```
wget http://nuhep.northwestern.edu/~ksung/YUGE/yuge_overlay.tgz tar -xf yuge_overlay.tgz
```

Create the Vivado project, then perform synthesis, implementation, and generate bit file.

```
ipbb proj create vivado yuge_c2c emp-fwk:projects/examples/yuge_c2c -t top.dep cd proj/yuge_c2c/ ipbb vivado project ipbb vivado synth -j4 impl -j4 ipbb vivado package
```

A copy of the output bit file can be found at `package/src/top.bit`.

- If needed, set up the Xilinx Virtual Cable on the YUGE

```
/etc/init.d/xvcServer-init stop cat system_top_6089_101_revA_jtag.bit > /dev/xdevcfg /etc/init.d/xvcServer-init start
```

- Load the firmware on the KU 115. Open the Hardware Manager in Vivado 2018.2 and program the KU 115. Note the IP address of the KU 115 is 192.168.200.16.

## uHAL software example

A uHAL code for testing data transmission and reception through the QSFP can be found here. The example sends word patterns to the transmit (tx) buffer and captures snapshots of the receive (rx) buffer, and can operate in loopback mode at each transceiver, or communicate between the two. The directory contains the `Makefile`, C++ source file, and connection XML. In order to compile, the `ipbus-software` package for cross

compilation (see above) must also be present.

Copy the executable and XML to the ZYNQ on the YUGE, e.g. (assuming the directory `/home/root/yuge_c2c_sw/emp/` has been created)

```
scp yuge_2quad_text.exe yuge:/home/root/yuge_c2c_sw/emp/ scp emp_yuge_connection.xml
yuge:/home/root/yuge_c2c_sw/emp/
```

Other files will also need to be copied to the YUGE in the same directory as the executable:

- the following XML files (paths relative to the IPBB work area)

```
src/emp-fwk/components/utils/addr_table/top_emp_slim.xml
src/emp-fwk/components/info/addr_table/emp_info.xml
src/emp-fwk/components/ctrl/addr_table/emp_ctrl.xml
src/mp7/components/mp7_ttc/addr_table/mp7_ttc.xml
src/emp-fwk/components/datapath/addr_table/emp_datapath.xml
src/emp-fwk/components/payload/addr_table/emp_payload.xml
src/mp7/components/mp7_ttc/addr_table/state_history.xml
src/mp7/components/mp7_datapath/addr_table/align_mon.xml
src/emp-fwk/components/datapath/addr_table/emp_region.xml
src/mp7/components/mp7_formatter/addr_table/mp7_formatter.xml
src/emp-fwk/components/links/be_mgt/interface/addr_table/emp_be_mgt.xml
src/mp7/components/mp7_links/addr_table/mp7_drp_chan.xml
src/mp7/components/mp7_links/addr_table/mp7_drp_common.xml
```

- the following shared object libraries

```
libboost_program_options.so libboost_program_options.so.1.64.0 libboost_timer.so
libboost_timer.so.1.64.0 libboost_unit_test_framework.so
libboost_unit_test_framework.so.1.64.0 libcactus_uhal_tests.so libcactus_uhal_tests.so.2.6
```

Log onto the ZYNQ on the YUGE from a PC.

- To enable chip-to-chip communication, the corresponding reset must be toggled off and on (not just turned on). To be safe, issue the following write commands on the memory addressed mapped to the reset (i.e. turn on, turn off, turn on). If chip-to-chip is not enabled when the uHAL software is run, the board will hang and must be rebooted.

```
poke 0x40020008 0x3 poke 0x40020008 0x0 poke 0x40020008 0x3
```

Run the executable in per quad loopback mode (argument 0) or quad-to-quad mode (argument 1):

```
./yuge_2quad_test.exe 1
```

-- KevinSung - 2019-07-15

---

This topic: Sandbox > YugeNorthwestern
Topic revision: r11 - 2019-09-25 - KevinSung