Qualcomm Research

Heterogeneous Multi-Core Architecture Support for Dronecode

Mark Charlebois, March 24th 2015

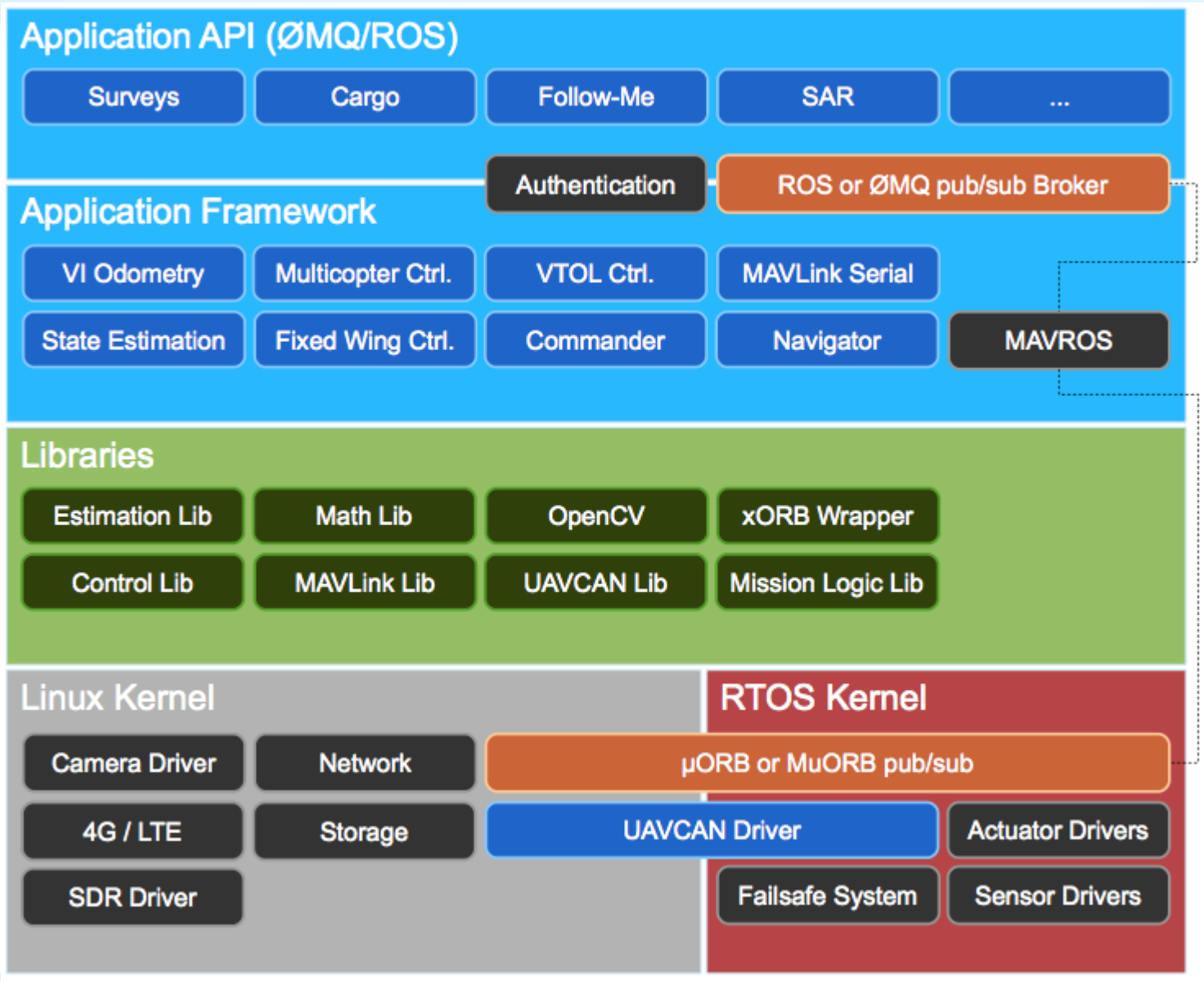
QUALCOMM®

Dronecode

- Qualcomm Technologies Inc (QTI) is a Silver member of Dronecode
- Dronecode has 2 main projects:

- https://www.dronecode.org/software/where-dronecode-used

PX4 SW Stack



https://pixhawk.org/_detail/dev/px4_software_architecture.png?id=dev:software_architecture

PX4 Software Architecture

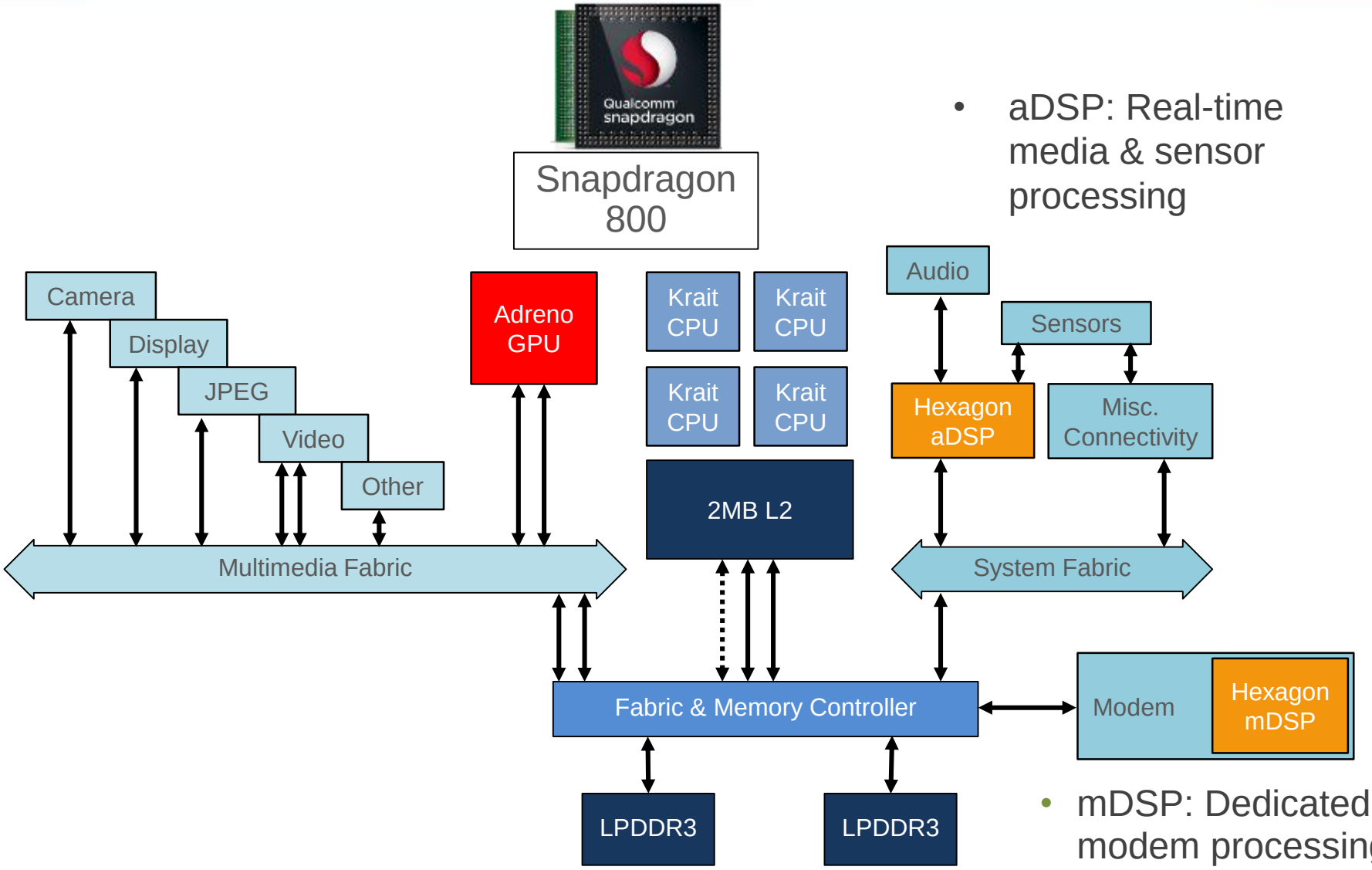
- Today PX4 Firmware is based on NuttX
- NuttX supports:
  - Single CPU
  - Flat memory model
  - Tasks
- PX4 Firmware uses devices for task synchronization
  - Custom device drivers (ioctl, read, write, poll, …)
  - Uses internal kernel structure data

PX4 on Snapdragon™ 600 SoC

Snapdragon 600 SoC

- The APQ8064 SoC has a heterogeneous multi-core architecture
  - Multi-core apps processor (4 Krait™ cores)
    - Linux™
  - SMP Hexagon™ processor
    - HW seen as 3 CPUs
    - Runs an RTOS (QuRT™)
    - Single process, multiple threads
    - Can run Linux
    - Supported in upstream kernel
  - Hexagon SDK provides a way to run SW on Hexagon
    - http://www.slideshare.net/QualcommDeveloperNetwork/21-hexagon-sdkmay919gg23
    - Developing DSPAL layer for POSIX API

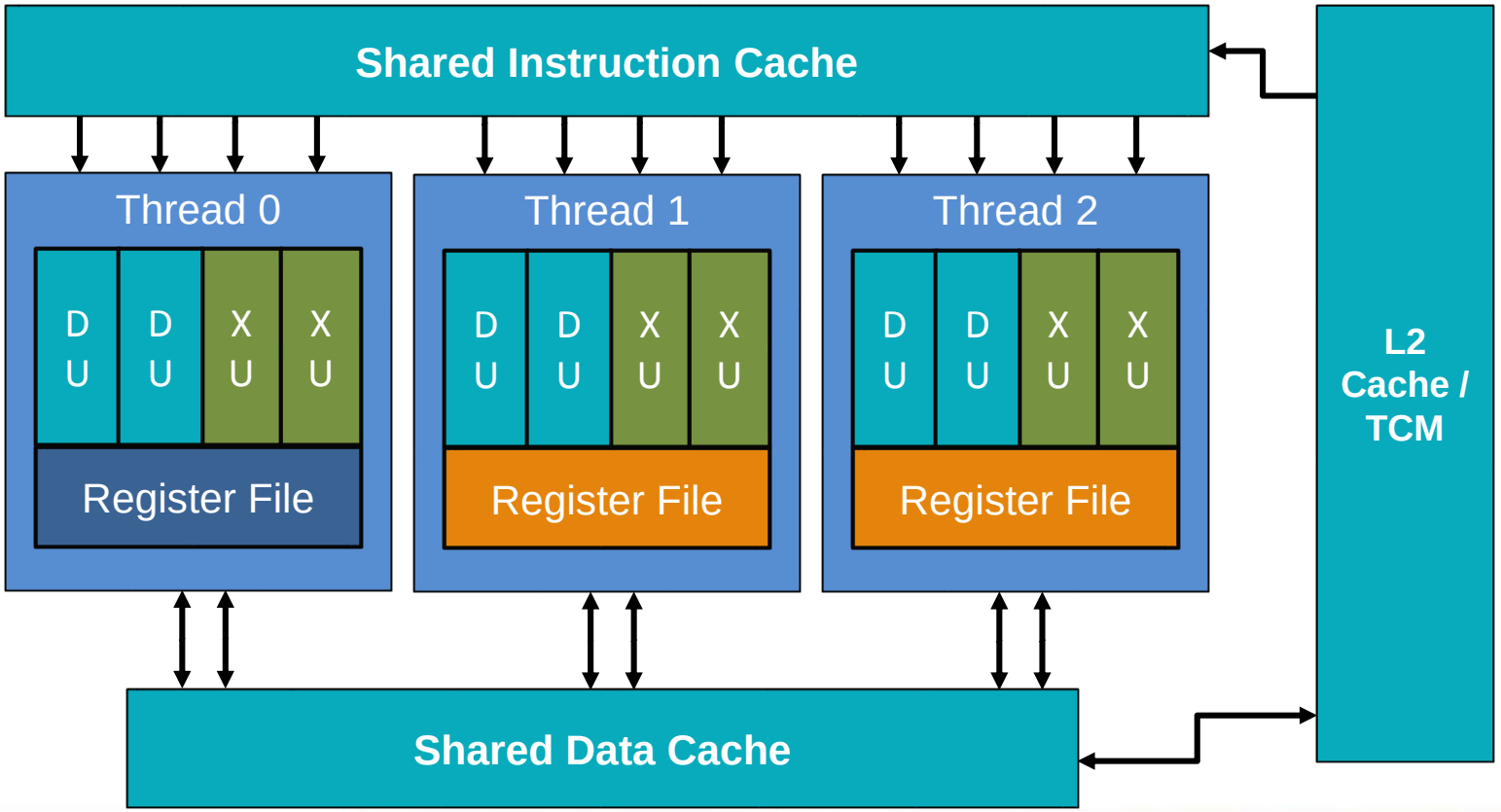
Hexagon™ DSP Processors in Snapdragon Products



- aDSP: Real-time media & sensor processing
- mDSP: Dedicated modem processing

Programmer's View of Hexagon DSP HW Multi-threading

- Hexagon V5 includes three hardware threads
- Architected to look like a multi-core with communication through shared memory



PX4 on Hexagon

- QuRT for realtime
- Hexagon SDK used to port PX4

- Select files from old PX4 version used for initial port
- Demonstrated flights of drones with PX4 based SW on Hexagon

- Hexagon support in progress for upstream PX4
  - DSPAL POSIX layer in development
  - Requires support for thread based PX4 build

PX4 Firmware Porting

Codebase Issues

- NuttX dependency, some code able to run under ROS
  - Looking at creating clean backend separation
  - Single CPU RTOS
  - Lots of use of internal kernel data structures
  - Tasks vs threads
  - err, errx, exit(), _exit(), main
- Param
  - NuttX uses memory segment and linker
  - Unit test creates static array
  - Difficult to split code across processors

Codebase Issues

- Time as uint64_t
  - uint64_t varies per platform
  - unsigned long on x86_64, unsigned long long on ARMv7hf/Krait
- Eigen
  - Lots of C++ issues
  - Is FLENS an option?
    - (http://apfel.mathematik.uni-ulm.de/~lehn/FLENS/index.html)
- Device support
  - Userspace device control vs kernel
    - I2C, SPI, UART

Thread Based Port of PX4

- Created a fork of PX4/Firmware on Github
  - https://github.com/mcharleb/Firmware
- Linux port of PX4/Firmware
  - Intermediate step
  - Single process, multiple threads, POSIX, user space
  - Enables definition of abstraction layer
  - Faster way to develop and test code
  - Can be done in parallel with DSPAL work

Top Level Code Changes

- makefiles/
  - firmware_linux.mk
  - firmware_nuttx.mk
  - toolchain_native.mk
    - Use clang or gcc (tested clang 3.4, 3.5 and gcc 4.8, 4.9)
  - setup.mk
    - PX4_TARGET_OS (nuttx, linux)
  - linux_elf.mk (create mainapp)
  - module.mk
    - -DPX4_MAIN=$(MODULE_COMMAND)_app_main
- tools/
  - linux_apps.py (create list of built-in “apps”)

Board and Config files

- Moved to subdirs for each OS
- nuttx/
  - NuttX board and config files
- linux/
  - Linux board and config files

Code Change Highlights

- Minimal code change to track upstream
  - Added abstraction headers
- src/platform
  - px4_posix.h, px4_tasks.h, px4_defines.h, etc
- Added implementation directories
  - src/platform/nuttx
  - src/platform/linux
- Created basic shell to instantiate “apps” under Linux
  - Similar to NuttX shell
  - Runs built-in “apps” using app_main(argc, *argv[])

Code Change Highlights

- Virtual device used to maintain use of ioctl calls
  - Modified Cdev → VCDev
- px4_open(“/dev/foo”)
  - devmap[“/dev/foo”]->vcdev->dev_open(px4_dev_handle_t *h)
- Split backends where required
  - foo_nuttx.cpp, foo_linux.cpp
- Converted process terminating calls
  - err, errx, exit, _exit

Created Demo/Test Apps

- src/platform/linux/tests
  - hello, hrt_test, vcdev_test
  - int PX4_MAIN(int argc, char **argv) { … }
- Use socat to create ttyS0 for mavlink
- Finding lots of failure cases (i.e. memset of nullptr)

The Future

Work To Do

- Add support for reading rc.S init in Linux port
- Finish DSPAL
  - Need PX4 spec for user space control of I2C, SPI
- Upstream QuRT/DSPAL port
  - Added support for missing mathlib support
    - Vector, Matrix, Quaternion, isinfinite…
  - Needed PX4 Param solution for Heterogeneous CPU usage
  - Debugging
    - Integrated with Qualcomm Diag framework
  - Sensor, PWM drivers
- uORB/MuORB
  - DDS?

Dronecode Future

PX4/APM	PX4/APM/ROS	PX4/APM
NuttX	Linux	DSPAL/QuRT
Cortex m3/m4	x86_64/Krait/ ARMv7hf	Hexagon

research.qualcomm.com