Porting GNU Radio to Multicore DSP+ARM SoC

A Purely Open Source Approach

Shenghou Ma, Vuk Marojevic, and Jeffrey H. Reed, Virginia Tech
Philip Balister, OpenSDR

Wireless@Virginia Tech

Who Am I?

- Maintainer of meta-sdr OpenEmbedded layer
- Work on products such as the USRP-E100
- GSoC mentor for embedded GNU Radio
- Helps develop enabling technology for hacking the IoT!

Motivation

- Vendor-specific development tools are only available on PC-based platforms
- Little influence from research community
- SDR is influenced by open source
- Open-source used in research and education

Contents

Trends in Wireless

- Coexistence and Interoperability
- Spectrum Sharing
- High Complexity and Capacity Demand for Radio Interface *and* Backhaul
- Infrastructure/Computing Sharing/Outsourcing

→ need development environment leveraging creativity, efficiency, adaptability and scalability

Wireless Development Issues

- Algorithm development cost (low level of outsourcing)
- Hardware architecture dependency (new computing demands subject to vendor upgrades)
- Low software and hardware reusability
- Long time-to-market for new wireless comm. systems
- Short mobile terminal life-cycle

Software-Defined Radio

Digital signal processing application that defines the radio functionality of a transceiver as software running on reconfigurable hardware

Hard real-time processing

Tradeoff between flexibility and processing performance and power consumption →

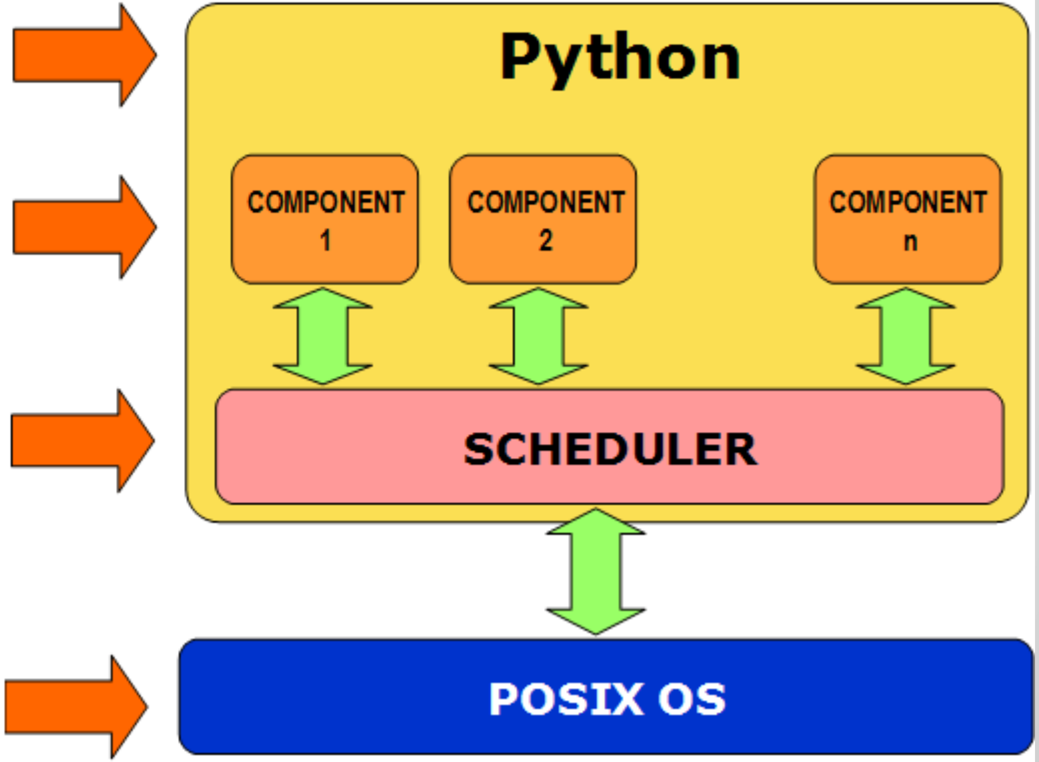
Heterogeneous Multiprocessing

GNU Radio - A Community Resource

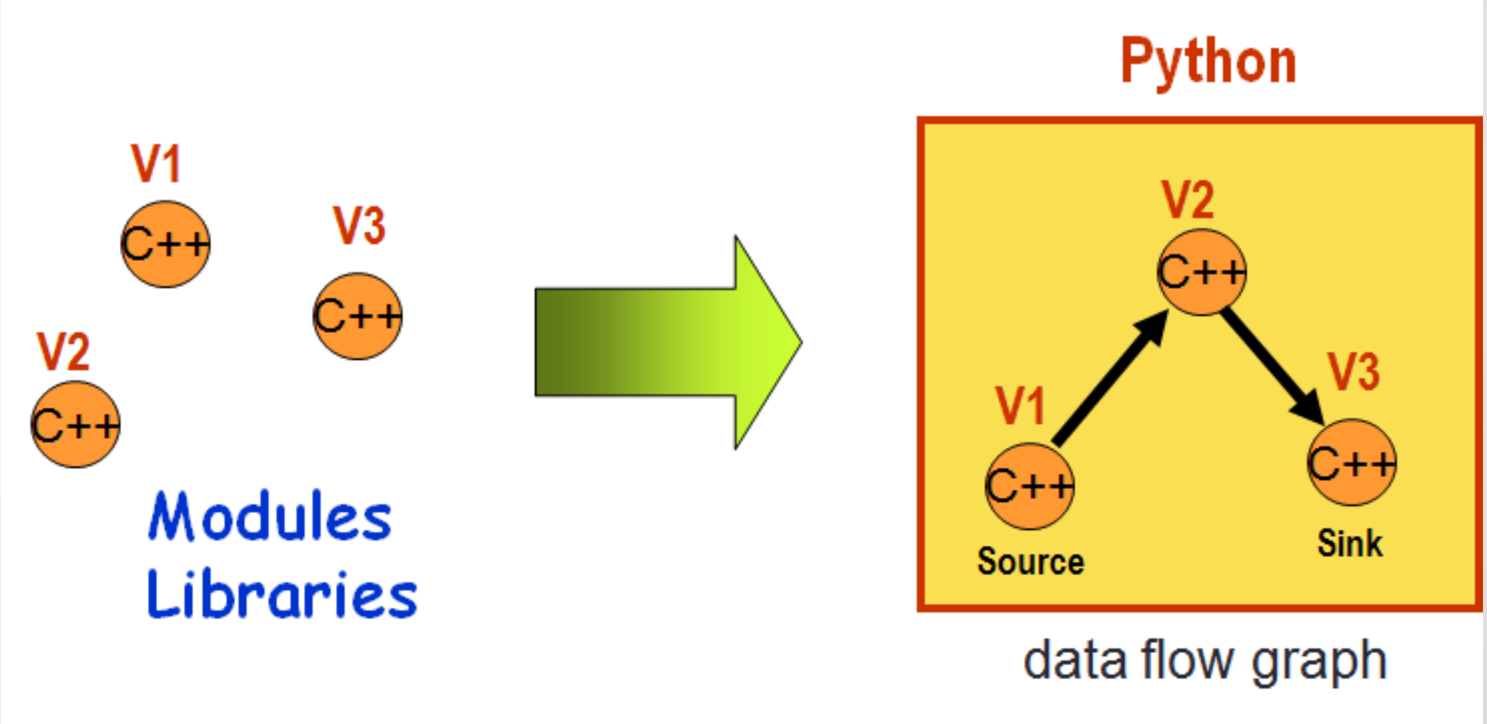
- **Free software development tools for building software radios**, http://gnuradio.org
- **Environment for rapid prototyping and testing** of SDR modules, waveforms, resource management algorithms, cognitive radio methods, ...
- Library of configurable SDR modules, signal source, measurement and visualization tools, ...
- Provide access to RF-front ends

Software Environment

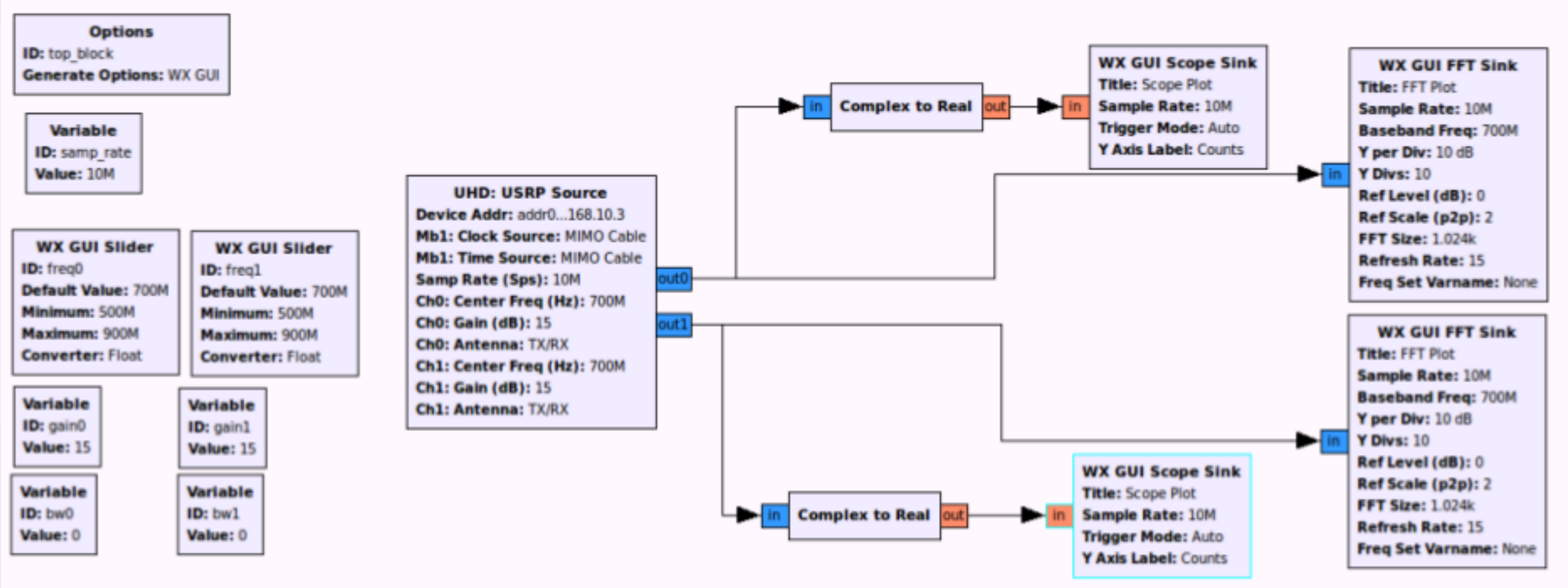
- Python script to control the creation of the waveform and its implementation
- Signal processing components typically programmed in C++
- Scheduler controlling the creation of threads and communication components
- UNIX/Linux-based Operating System



Modules and Graphs

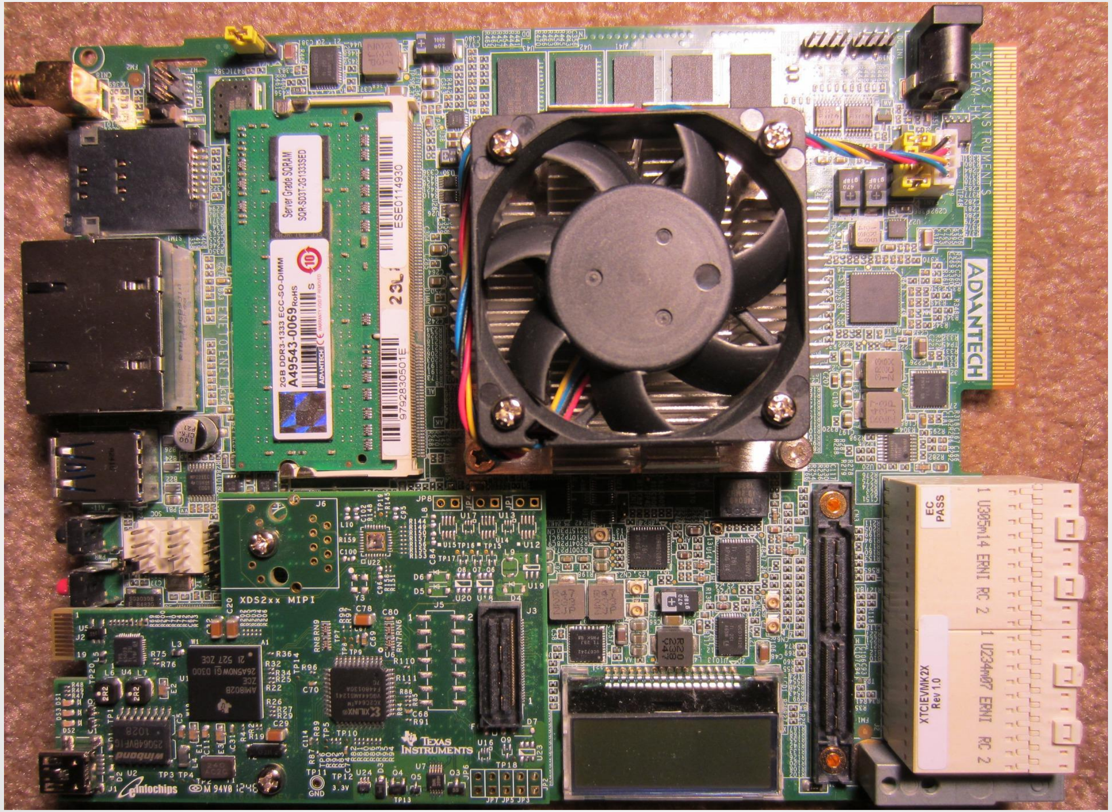


GNU Radio Companion (GRC)

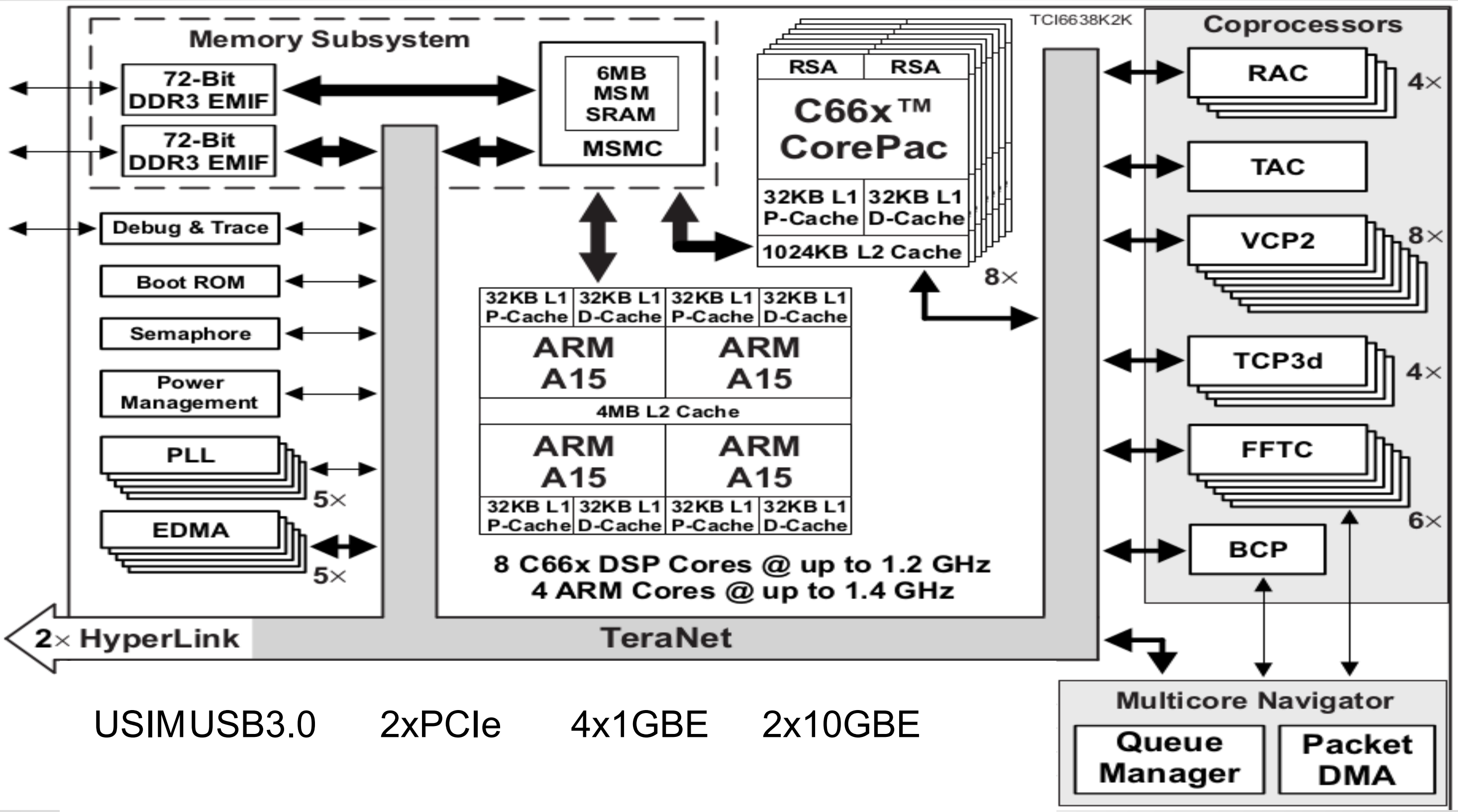


TI’s KeyStone II:
Multi-core ARM+DSP SoC

KeyStone II EVM



KeyStone II: TCI6638K2K



Major Components

- 4 ARMs Cortex A15 @ 1.4 GHz
- 8 C66x DSP cores @ 1.2 GHz:
  - 38.4 GMACs/Core or 19.2 GFLOPs/Core
- Communications accelerators
  - 2 FFT, 4 Turbo Decoders, 8 Viterbi Decoder, ...
- High-Speed Interfacing
- Multicore Navigator
- 6 MB Shared SRAM, 1 MB L2 Cache/RAM for each core

Vision and Related Work

Vision

Three milestones:

1. Distribute work to other machines with the same architecture.
2. Separate block processing code out and re-implement in a common language
3. Unified Scheduler for heterogeneous architectures!

Related Work

- Al Fayez’s gr-DSP (C64x, DSPBIOS)
- RSP2011: Applying GPU to SDR
- FPL2013: GReasy (GNU Radio blocks on FPGA)
- GRCon2013: Using DSPs in GNU Radio

Digital Signal Processor for SDR

...Education, Research and Rapid Waveform Development and Deployment

DSP as accelerators

DSP as complete systems

Rapid prototyping

Rapid reconfiguration → CR, DSA, ...

open-source era … sharing … community support

Proprietary software tools

Linux on embedded ARMs

GNU Radio on embedded ARMs

Integrated execution environment

Eventually...

System Analysis, Evaluation and Contribution

Memory Organization

- both ARM and DSP has 2 level of caches
- ARM caches could be flexibly configured as cache-only, SRAM-only or a mix of the two
- 6MB of shared memory on chip
- No consistency between DSP caches and between DSP caches and ARM caches

Communication Issues

Streaming vs. Shared Memory?

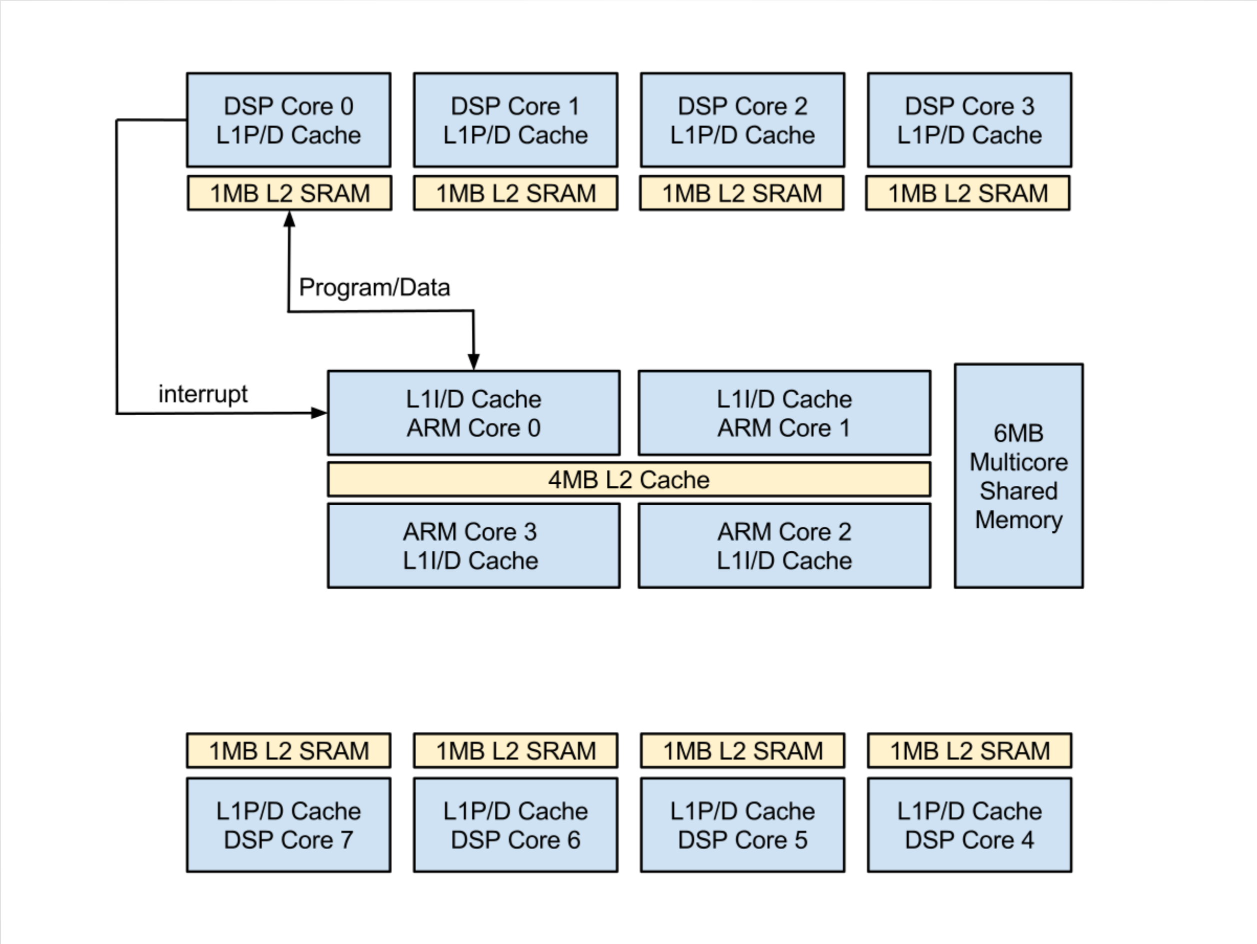
GRDSP and GRGPU both stream data over PCI-E (no other choice)

GReasy streams over Ethernet.

GR on Zynq in GSOC 2013 streams over AXI-Stream.

On the KeyStone II SoC, Sharing (on-chip) memory is perfectly fine and efficient.

System Architecture



More details on interactions

Linux has framework for co-processors (remoteproc) and it’s used in Android.

- Launch a hardware thread on to a coproc.
- use UIO (user-space IO) to get the inter-core interrupt (already supported in kernel for the board)

Scheduler

- lack of MMUs in DSP
- No cache coherence between DSP cores
- Need for centralized scheduler
- DSP cores as specialized threads
- Extendable to other accelerators (e.g. FPGAs)

DSP w/o Proprietary Software

Just a couple register writes, isn’t it?

The troubles:

- No visibility of what the DSP is doing (no debugger)
- Lack of detailed documentation, TI assumes you are using DSPBIOS and its Linux drivers
- Complex clock domains and power control

Evaluations

gcc 4.8.2 c674x backend is generally good, but:

```
float fir(int N, float coef[], float samples[]) {
    float r = 0.0f;
    int i;
    for (i = 0; i < N; i++)
        r += samples[i] * coef[i];
    return r;
}
```

```
fir:
        cmplt    .l1  0, A4, A0
    ||  shl  .s1  A4, 2, A3
    ||  sub  .d1
    [!A0]     b
    ||  sub  .d1
        shru .s1  A4, 2, A5
    ||  mvk  .d1  0, A4
        add  .d1  A5, 1, A1
        nop  3
    ;; condjump to .L4 occurs
.L3:
        ldw  .d2t1    *B4++[1], A7
    ||  sub  .d1  A1, 1, A1
        ldw  .d1t1    *++A6[1], A8
```

perfect candidate for software pipeline, but gcc doesn't support SPLOOP!

```
        nop  4
        mpysp     .m1  A7, A8, A9
        nop  1
[A1]    b    .s1  .L3
        nop  1
        addsp     .l1  A4, A9, A4
        nop  3
;; condjump to .L3 occurs
        ret  .s2  B3
        nop  5
;; return occurs
.L4:
        ret  .s2  B3
||      mvk  .d1  0, A4
        nop  5
;; return occurs
```

What is Available

- binutils-2.23.2/gcc-4.8.2 based C674 toolchain (compatible with C66x)
- (partial) libc and libm
- liquid-dsp for DSP cores, replaces TI's DSPLIB
- ARM lib to control DSP cores

http://github.com/GRDSP

What is in Progress

1. Remoteproc support for DSP cores.
2. Could we extend ORC/Volk to cover DSP?
3. Rewrite more blocks.
4. Support more platforms
    a. FPGAs
    b. embedded GPUs (e.g. Mali, and other vendors')

How to Contribute

URL: http://github.com/GRDSP

Create issues on the GitHub project.

Contact: Shenghou Ma, minux@vt.edu

Open-source contributions welcome

References

- A Standalone Package for Bringing Graphics Processor Acceleration to GNU Radio: GRGPU, http://gnuradio.org/redmine/ attachments/download/257/06-plishker-grc-grgpu.pdf, 2011.
- W. Plishker, G.F. Zaki, S.S. Bhattacharyya, C. Clancy, and J. Kuykendall, “Applying graphics processor acceleration in a software defined radio prototyping environment,” in 22nd IEEE International Symposium on Rapid System Prototyping (RSP), 2011, pp. 67–73.
- Using Digital Signal Processors in GNU Radio, GRCon 2013, http://www.trondeau.com/storage/grcon13_presentations/grcon13_ford_snl_dsps.pdf
- A. Love and P. Athanas, “Rapid modular assembly of Xilinx FPGA designs,” in 2013 23rd International Conference on Field Programmable Logic and Applications (FPL), 2013
- KeyStone Arch. Multicore Navigator, http://www.ti.com/lit/ug/sprugr9f/sprugr9f.pdf
- C66x DSP Cache User Guide, http://www.ti.com/lit/ug/sprugr9f/sprugr9f.pdf