Atomix:
Virtualization substrate for LTE Data & Control Planes

Manu Bansal, Aaron Schulman, Sachin Katti
Stanford University

ONRC Review
May 12, 2015
Expressing data-plane stacks as flow-graphs

**SoftRAN OS**: Platform to execute any combination of flow and control graphs expressed in high level programs on COTS distributed hardware while meeting latency & throughput requirements.
Virtualized data-plane stack

Virtualizing wireless data-plane stack requires ability to modify and move functions
Primitives needed for virtualizing data-planes

Base-station software needs to provide a modular interface to tap, tweak, and insert.
Current base-stations are programmable

```c
int main() {
    //PHY & MAC
    ...
}
```

Base-stations do not provide a modular interface to make modifications
Stacks require high throughput and low latency

High processing throughput

40Mps == 640Mbps

Data frame

ACK frame

250us

16us

Time

Low processing latency

WiFi stack example
(LTE has easier latency constraints)

Software must deliver hardware-like performance
Getting hardware-like performance

- Customized hardware
  - DSP0
  - Local SRAM
  - Shared SRAM
  - DSP1
  - Local SRAM

- Hand-optimized software

- Pipeline programming
- Memory management
- Inter-core data transfers
Getting hardware-like performance

- Customized hardware
  - Hand-optimized software
  - Pipeline programming
  - Memory management
  - Inter-core data transfers

Diagram:

- DSP0
  - Local SRAM
  - Shared SRAM

- DSP1
  - Local SRAM
Getting hardware-like performance

- DSP0
- DSP1
- Local SRAM
- Shared SRAM

Pipeline programming
Memory management
Inter-core data transfers
Customized hardware
Hand-optimized software

Modular changes can have unpredictable effects on timing
This Talk: The Atom Abstraction

Atom: A unit of execution with fixed, known timing

Composability: Composition of atoms is also an atom

In Atomix, base-station software...
1) Can be built entirely out of atoms
2) Achieves hardware-like performance
3) Enables modular modifications
Base-station software stacks can be built with atoms

Atomix stacks can meet throughput and latency

Atomix stacks are easily modified to new stacks
WiFi signal processing chain

Data flowgraph over signal processing blocks
Implementing blocks as atoms

1) Split out branches, 2) Fix data lengths
Make branching explicit
Splitting out flowgraph modes

Split out flowgraphs to make branching explicit
Implementing flowgraphs as atoms

Explicitly model data access cost using FIFO atoms
Parallelizing flowgraphs with atoms

Core 0

SYNC → F → CSI → F → BPSK Atom

<flowgraph>

Core 1

SYNC → F → OFDM → F → BPSK Atom

<flowgraph>
Parallelizing flowgraphs with atoms

Explicitly model data transfer cost as an atom
Implementing decisions with atoms

Make branch explicit, push to the top-level
Atomix framework in 1-slide

• **Everything as an atom**
  – Signal processing components
  – Hardware management components
• Atoms for blocks, flowgraphs, states
• Simple control flow makes atoms composable
• Declarative language allows easy modification

Atoms enable modularity, precise timing, efficient pipelines
Base-station software stacks can be built with atoms
Atomix stacks can meet throughput and latency
Atomix stacks are easily modified to new stacks
Fine-grained pipeline parallelism

80-sample buffers (OFDM symbols)

...  ■  ■  ■

DSP0
- SYNC
- CSI
- OFDM

DSP1
- EQ
- QAM64
- DEINTER-LEAVER
- DEPUNCTURER

DSP2
- DECODE-SCATTERER
- VITERBI-ISSUE
- DECODE-GATHERER

DSP3
- DESCRAMBLER
- CRC32

Decoded bits...
- 010110

VCP0  VCP1  VCP2  VCP3
Packet
Energy
CRC
Toggle
Fine-grained pipeline parallelism

Atomix WiFi decodes 10MHz with resources to spare
Tight packet decode latency

WiFi highest-MCS 1000-byte packets, CDF of decode latency
(deadline = 32us at 10MHz)

Atomix WiFi decodes 10MHz in low latency, predictable timing
Experience with WiFi in Atomix

- Signal processing functions (C)
- Schedule, Resource Assignment
- Parallelized Atoms (Ax)

3,000 Loc (with Atomix)

30,000 LoC (w/o Atomix)

Low-level code (C)

Atomix runtime libraries

Native app binary
Base-station software stacks can be built with atoms
Atomix stacks can meet throughput and latency
Atomix stacks are easily modified to new stacks
Location-aware stack (SecureArray, Mobicom’13)

4 new signal processing blocks
30 lines of code to add app
Predictable change in SYNC atom

No change in WiFi packet decode latency

Long-range links (WiLDNet, NSDI’07)

2 new signal processing blocks
20 lines of code to add app
Predictable change in SYNC, DATA
Related work

- Modular frameworks for GPPs and FPGAs
  - SORA: works on GPPs, no clear mapping to DSPs
  - AirBlue: Targets FPGAs, different challenges than DSPs
  - Ziria: complementary to Atomix

- Embedded real-time operating systems (Neutrino, VxWorks, TI SYS/BIOS)
  - Typically for low sample rate apps (e.g., anti-locking brakes)
  - Misfits for expressing modular signal-processing apps
  - No abstractions for blocks, flowgraphs, state-machine
Conclusion

Atomix: Framework for virtualizing base-station dataplane functions [Atomix, Usenix NSDI ’15]

- Everything is an atom
- Hardware-like performance
- Modularity to tap, tweak, insert

On going:
- Scheduling over a cluster of baseband processors (DSP/ARM, x86)
- Extending to L2-L7 NFV packet processing