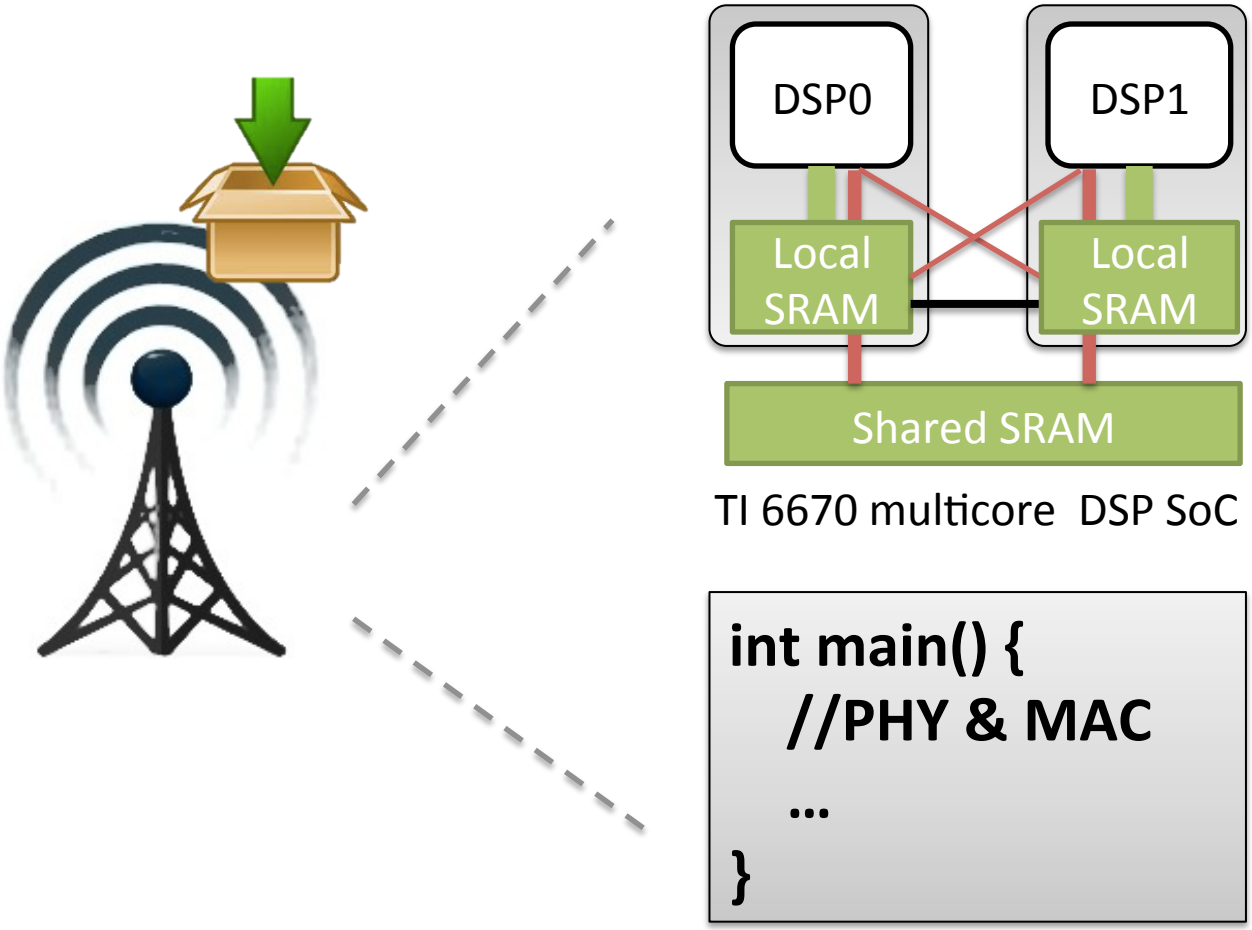
Atomix:
A Framework for Deploying Signal Processing Applications on Wireless Infrastructure

Manu Bansal, Aaron Schulman, Sachin Katti
Stanford University

NSDI '15

Software-defined base-stations



Wireless infrastructure is programmable

We could be deploying apps into infrastructure



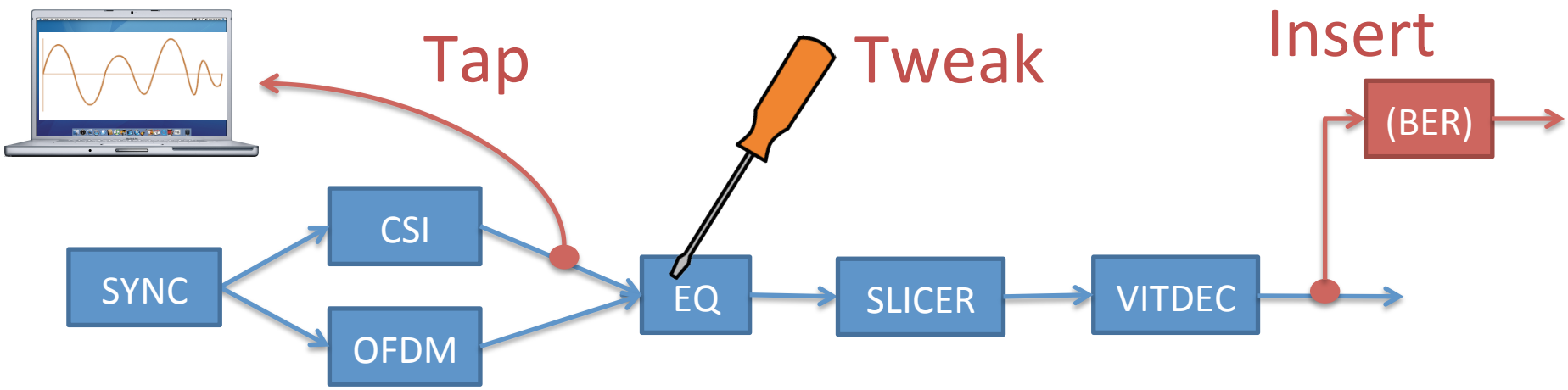
RF localization
(SecureArray)

Video-optimized
wireless stack
(APEX)

BER feedback
(SoftRate)

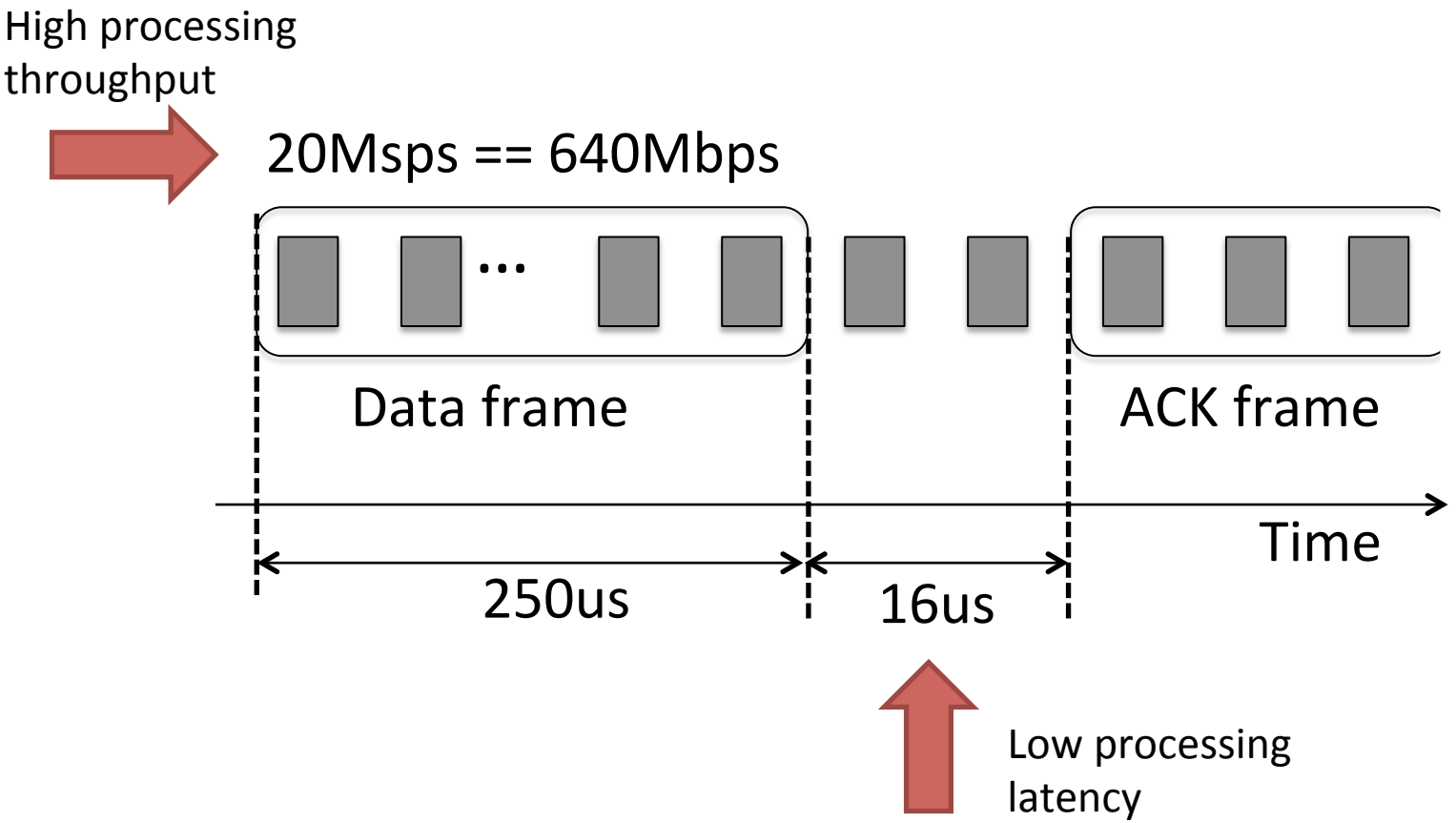
What modifications are needed to deploy apps?

Primitives needed for deploying apps



Base-station software needs to provide
a modular interface to tap, tweak, and insert

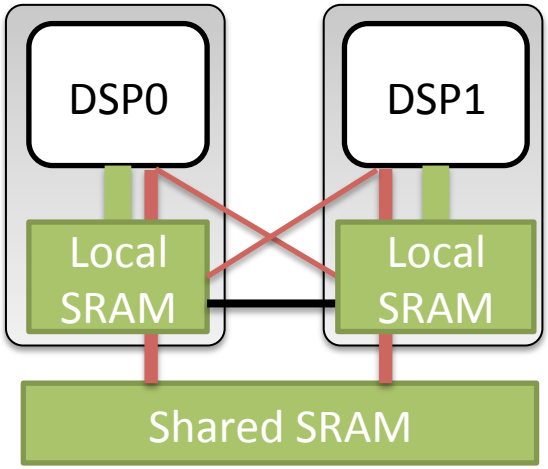
Apps require high throughput and low latency



WiFi stack example
(LTE has easier latency constraints)

Software must deliver hardware-like performance

Getting hardware-like performance



Pipeline programming

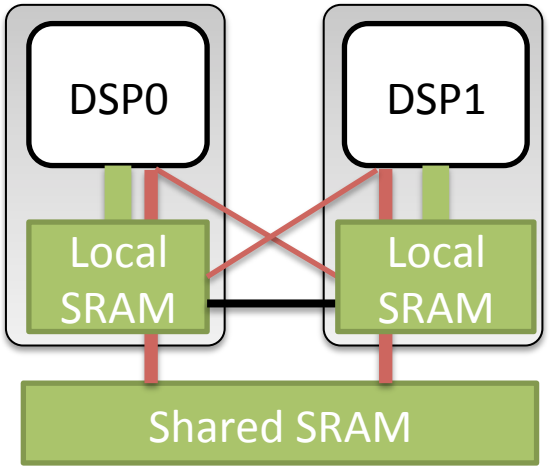
Memory management

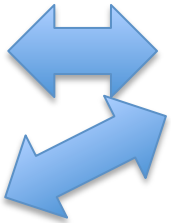
Inter-core data transfers

Customized hardware

Hand-optimized software

Getting hardware-like performance





Pipeline programming

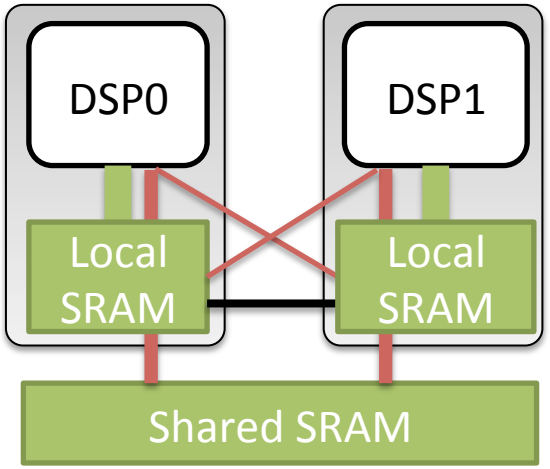
Memory management

Inter-core data transfers

Customized hardware

Hand-optimized software

Getting hardware-like performance





Pipeline programming

Memory management

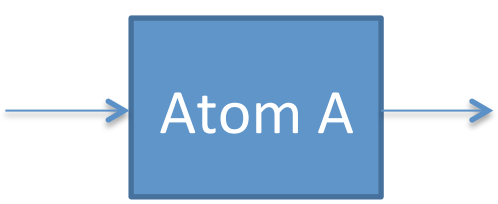
Inter-core data transfers

Customized hardware

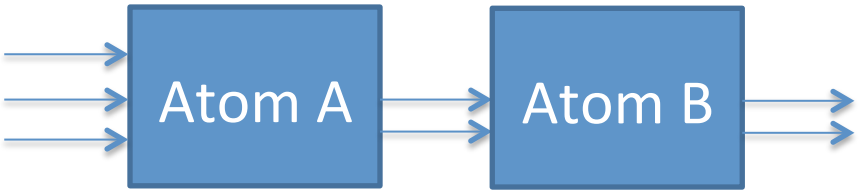
Hand-optimized software

Modular changes can have unpredictable effects on timing

The Atom Abstraction



t_A (e.g. 200 cycles)



$t_A + t_B$

Atom: A unit of execution with fixed, known timing

Composability: Composition of atoms is also an atom

Base-station software in Atomix…

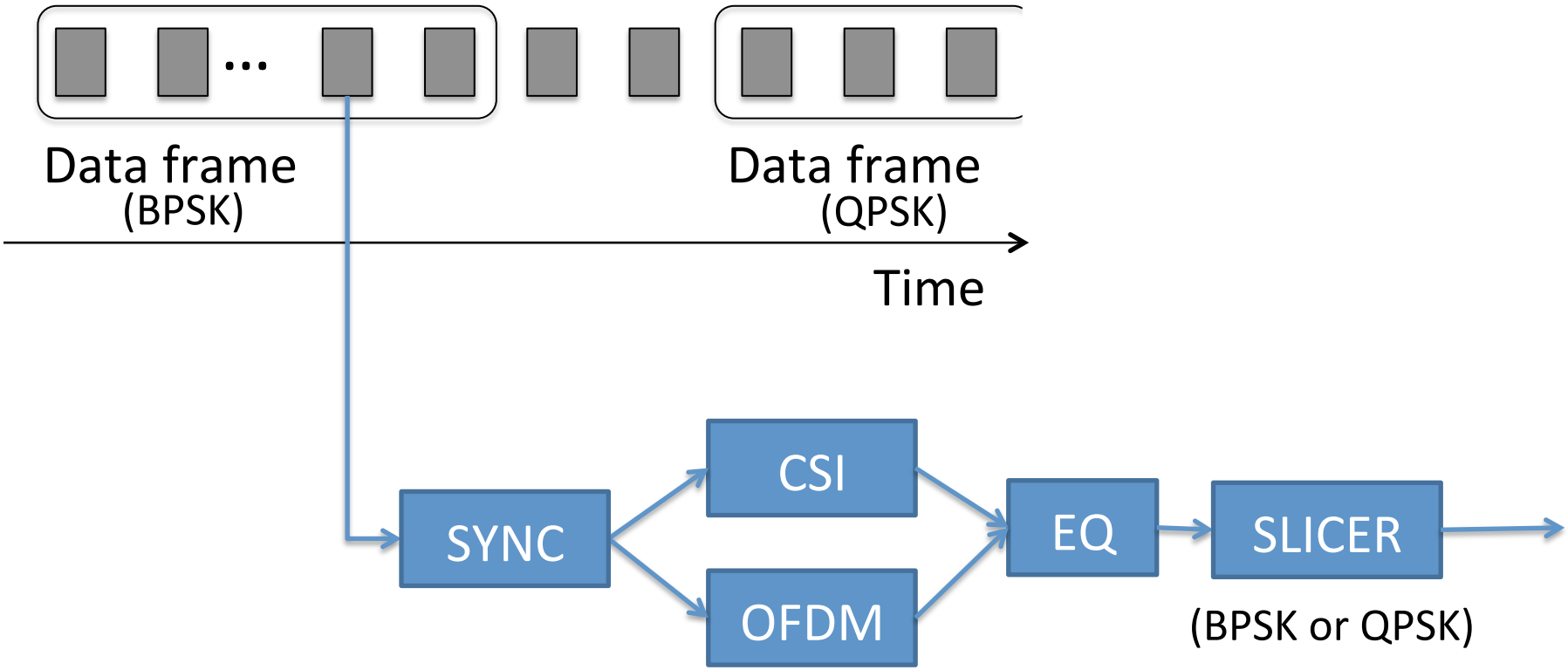
1) Can be built entirely out of atoms
2) Achieves hardware-like performance
3) Enables modular modifications

Base-station software can be built with atoms

Atomix WiFi chain can meet throughput and latency

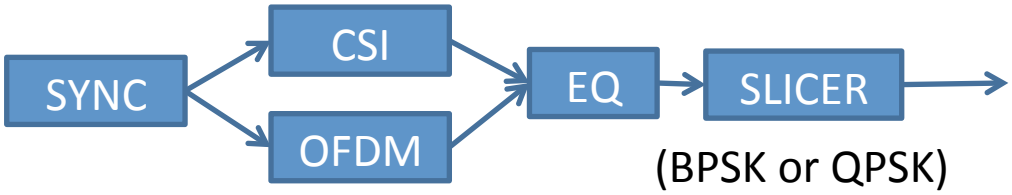
Apps are easily added to Atomix WiFi

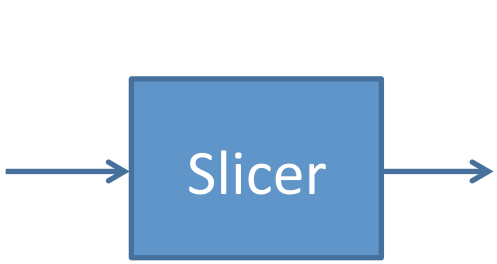
WiFi signal processing chain



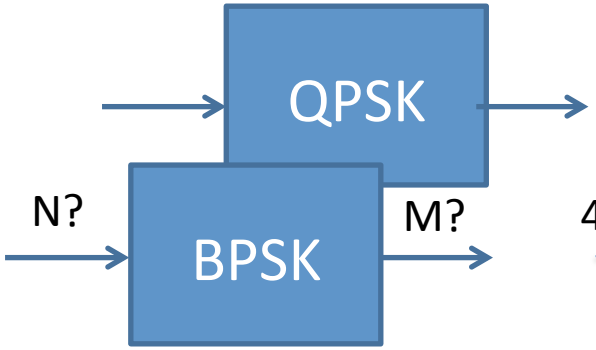
Data flowgraph over signal processing blocks

Implementing blocks as atoms

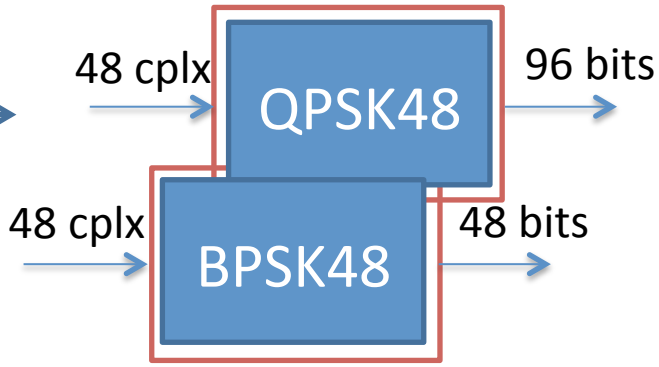




Constellation?
BPSK: 200cy
QPSK: 400cy



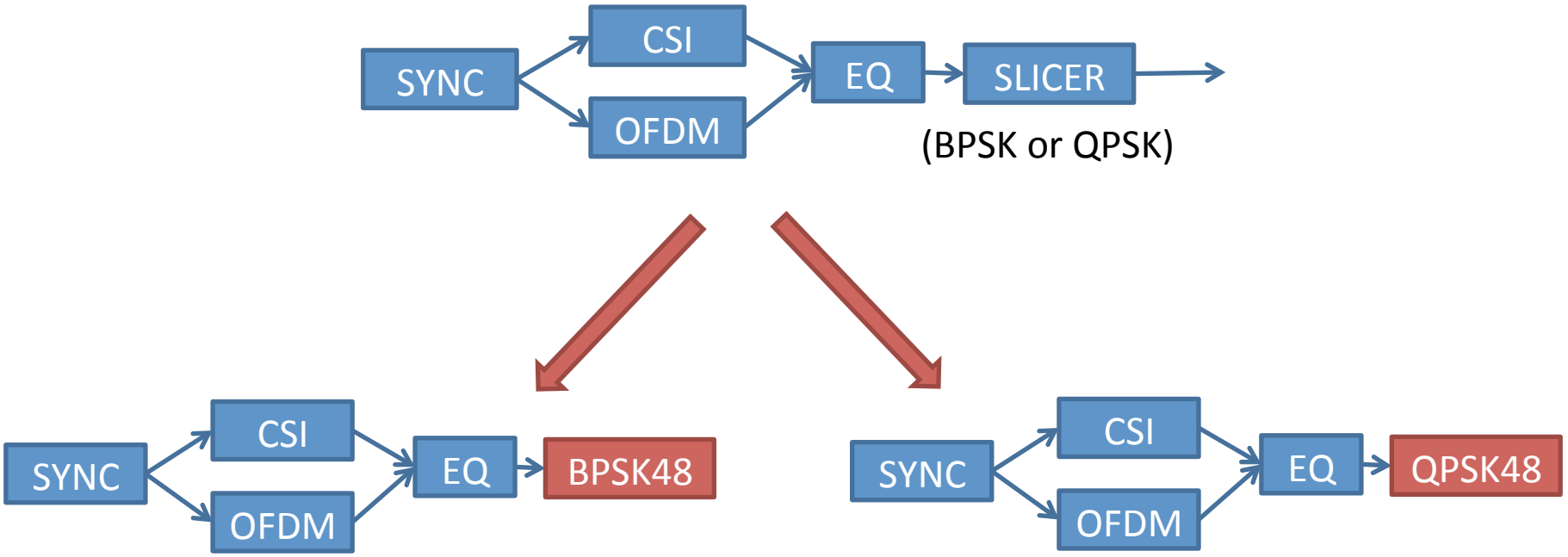
Input data length?
1 OFDM symbol: 200cy
2 OFDM symbols: 400cy



$t_{BPSK48} = 200cy$

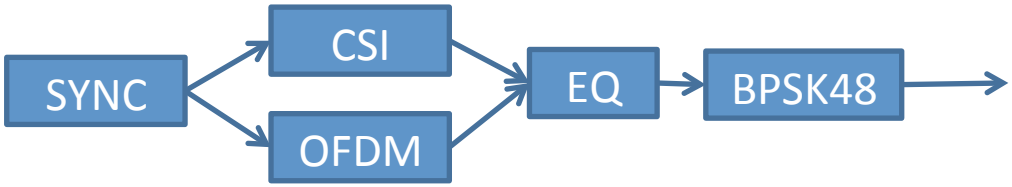
1) Split out branches, 2) Fix data lengths
Make branching explicit

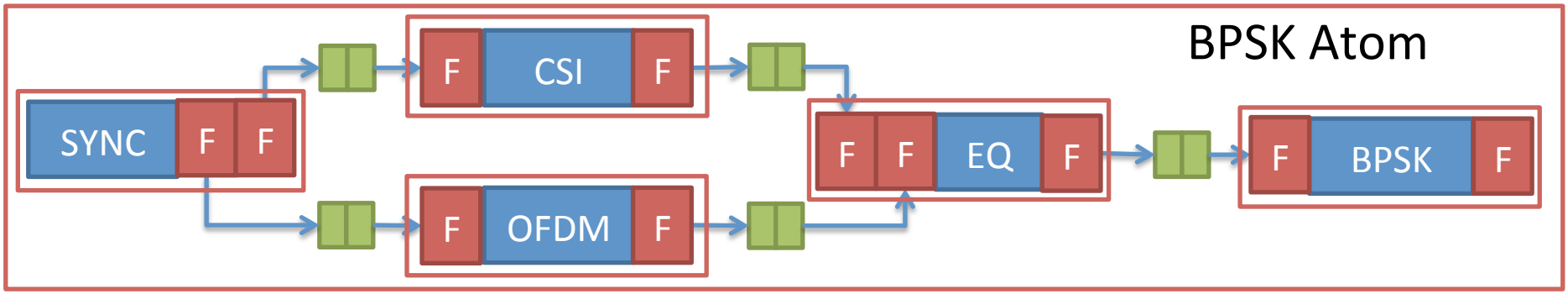
Implementing flowgraphs as atoms



Make branching explicit

Implementing flowgraphs as atoms

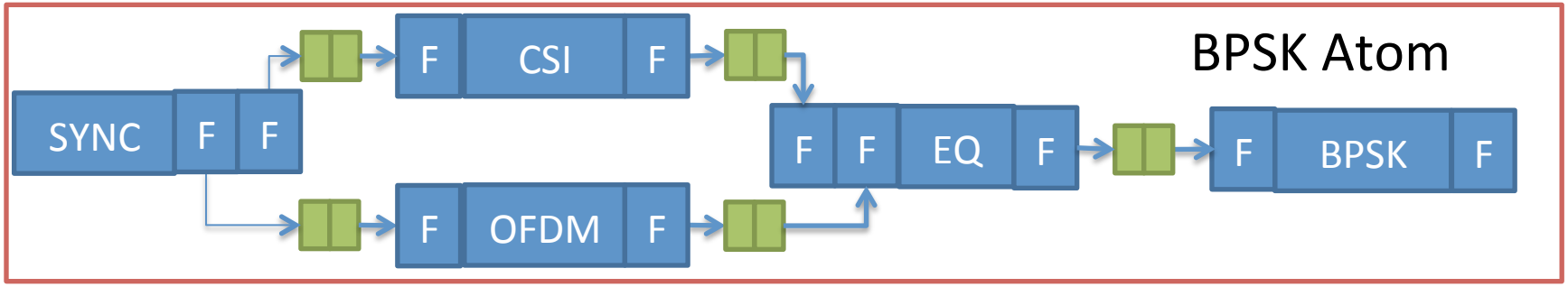




Explicitly model data access cost using FIFO atoms

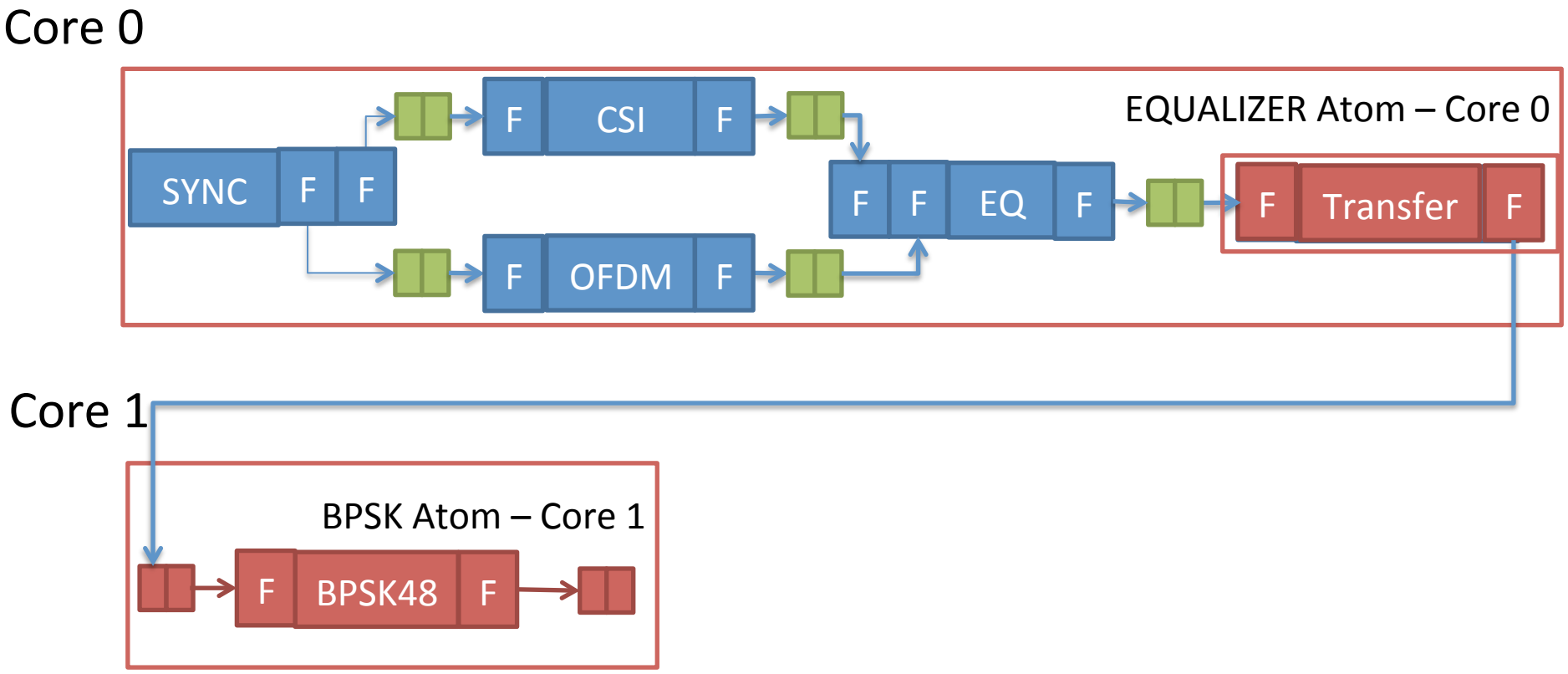
Parallelizing flowgraphs with atoms

Core 0



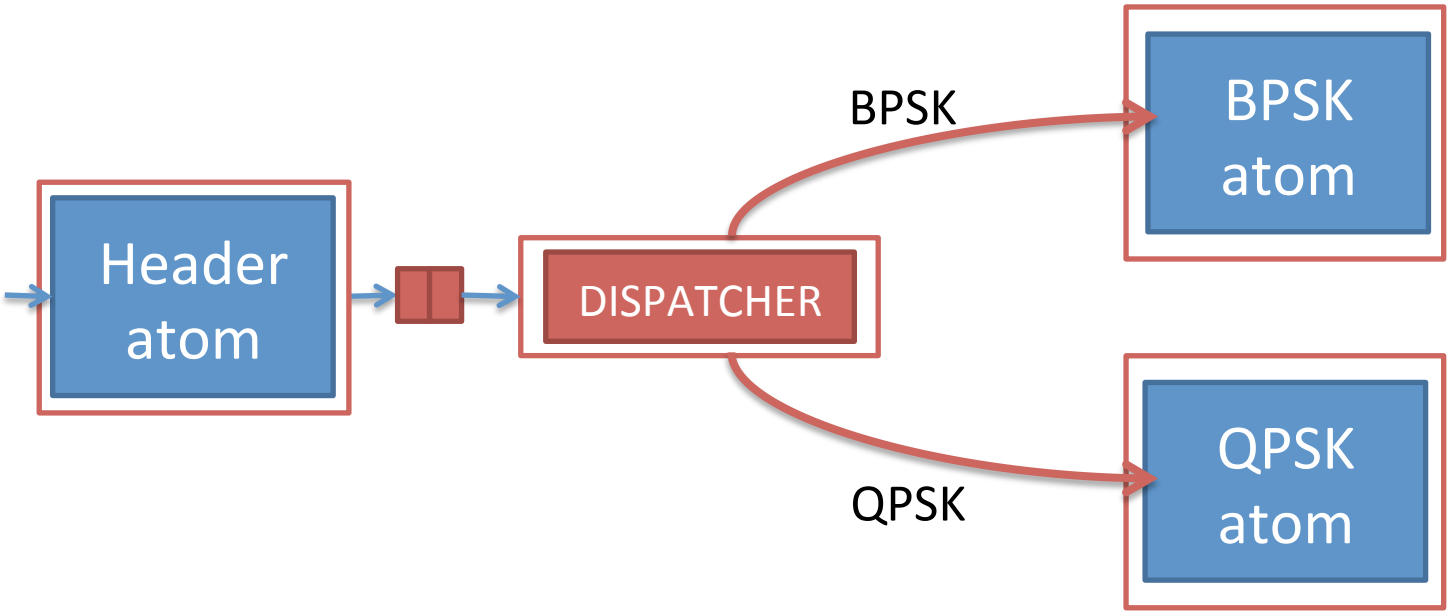
Core 1

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

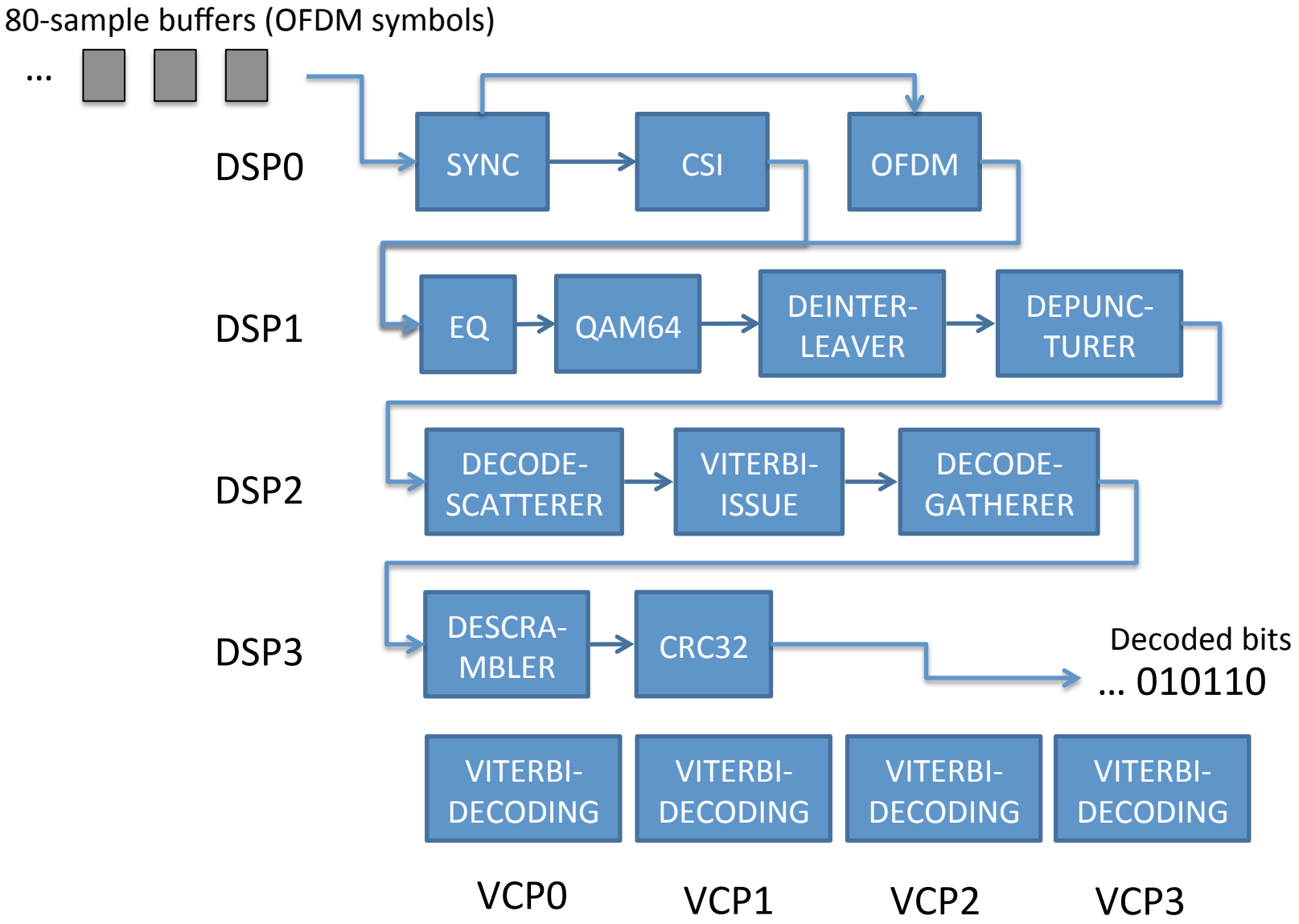
Atoms enable modularity, precise timing, efficient pipelines

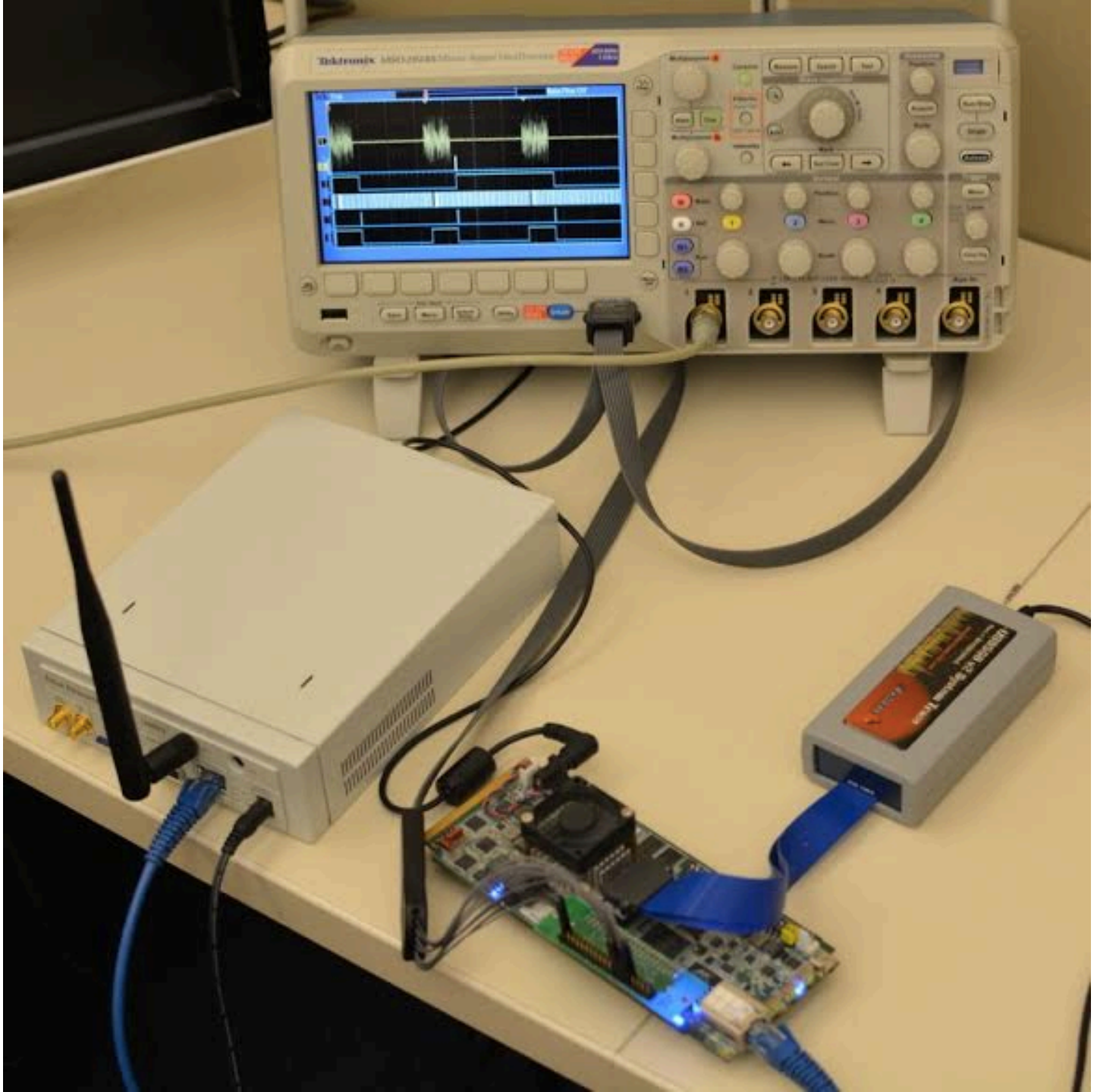
Base-station software can be built with atoms

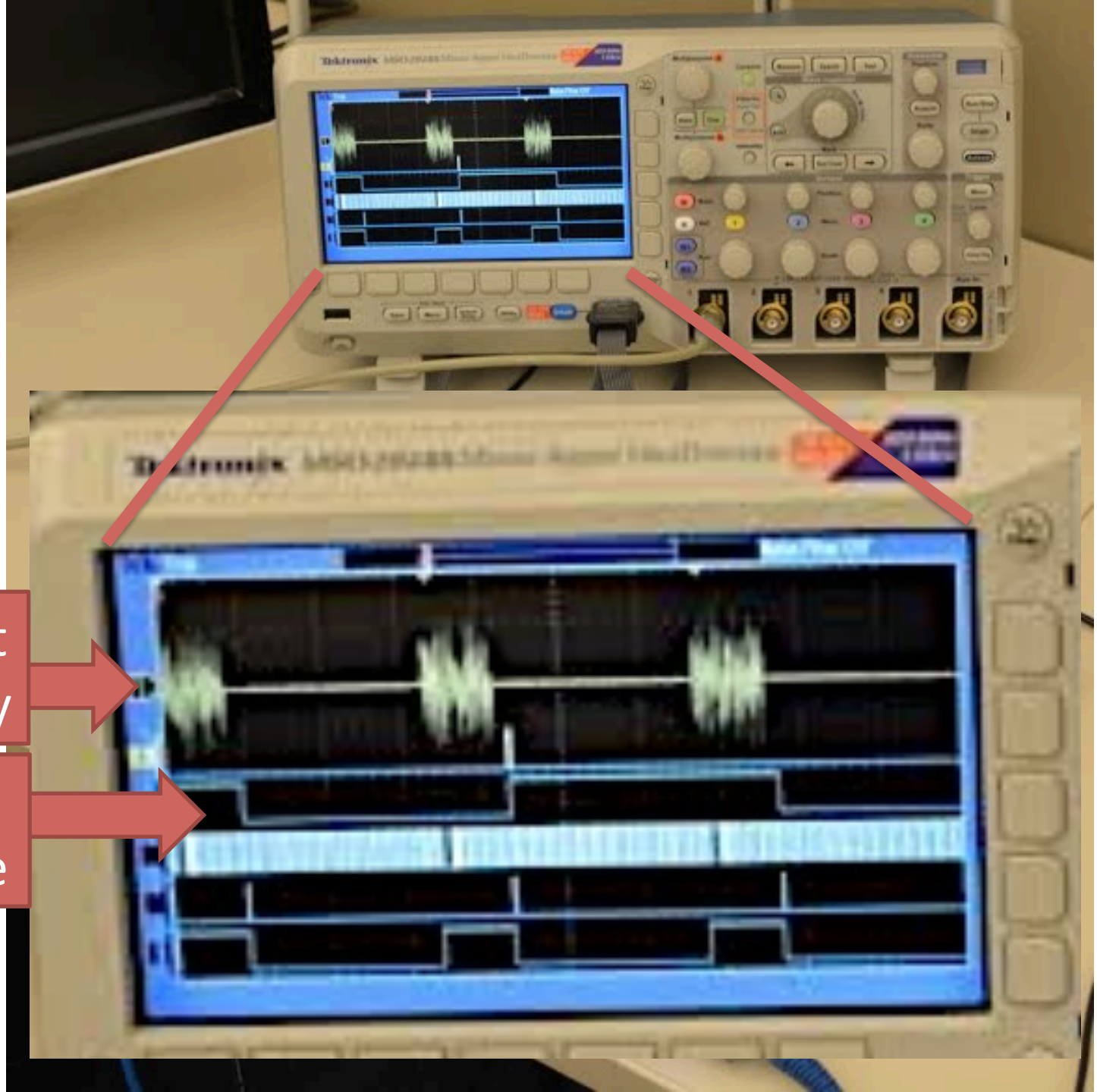
Atomix WiFi chain can meet throughput and latency

Apps are easily added to Atomix WiFi

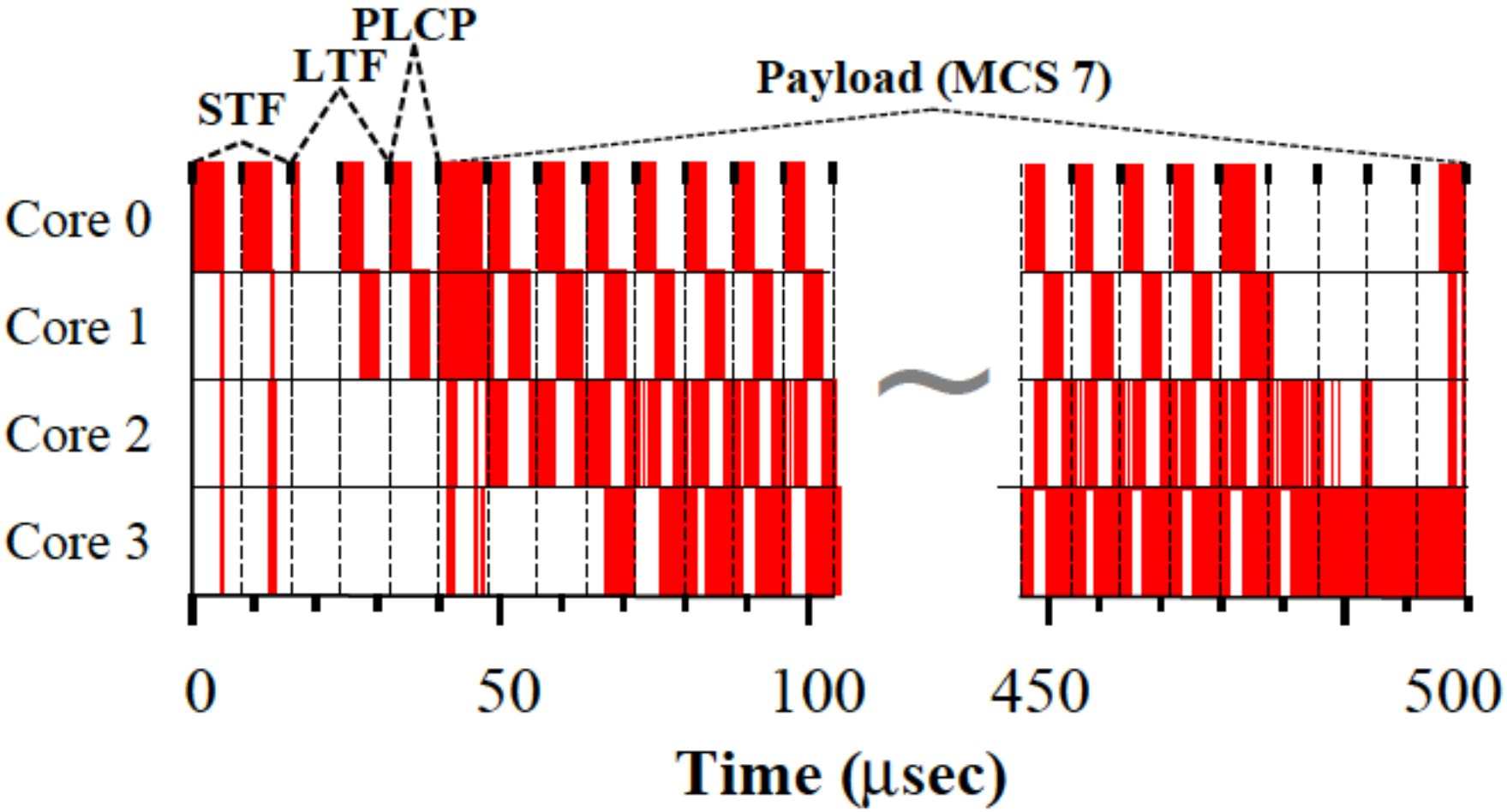
Fine-grained pipeline parallelism




Tektronix

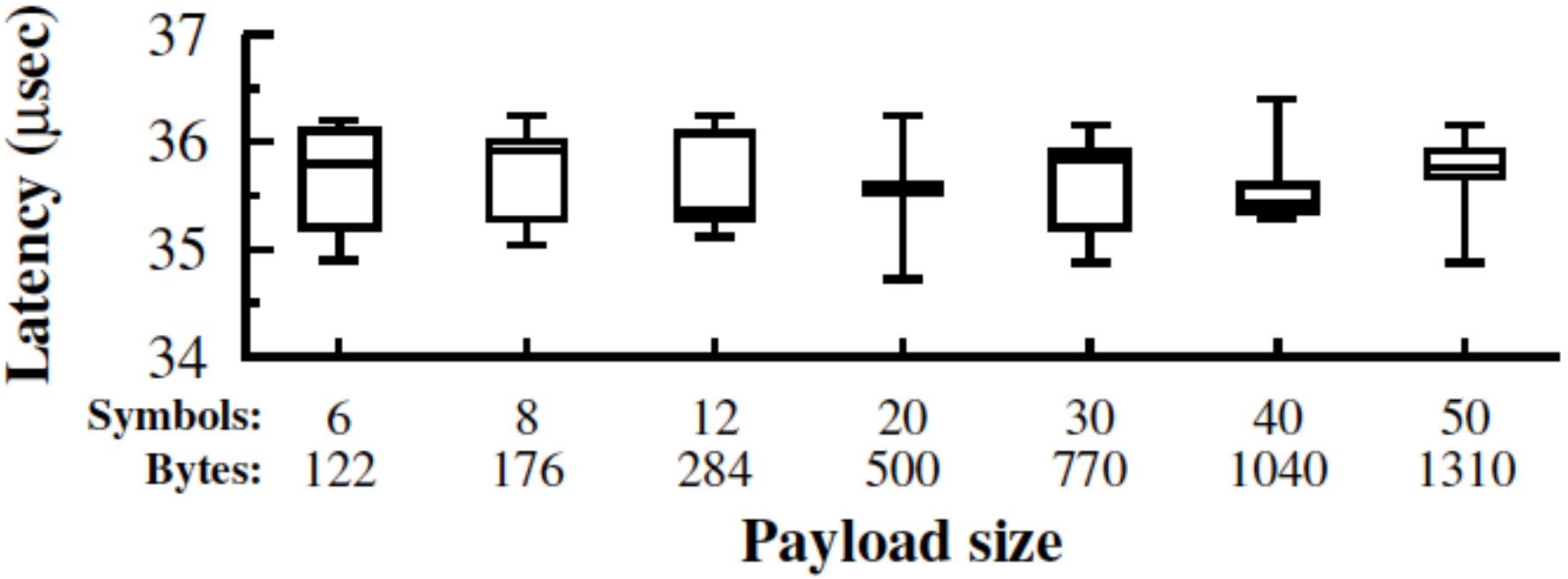

Packet Energy
CRC Toggle

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 = 64us at 5MHz, 32us at 10MHz, 16us at 20MHz)

Atomix WiFi decodes 10MHz in low latency, predictable timing

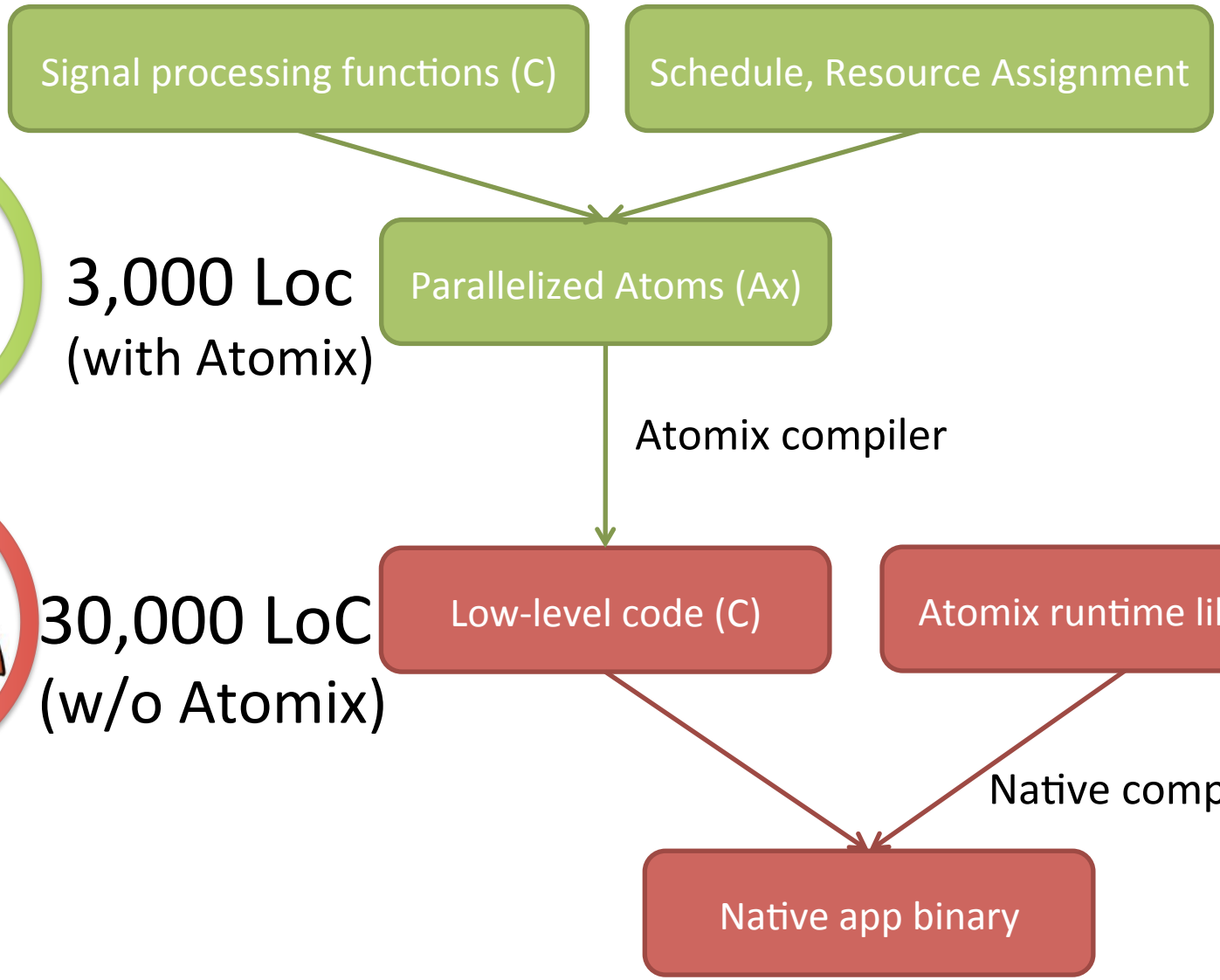
Experience with WiFi in Atomix



3,000 Loc
(with Atomix)



30,000 LoC
(w/o Atomix)

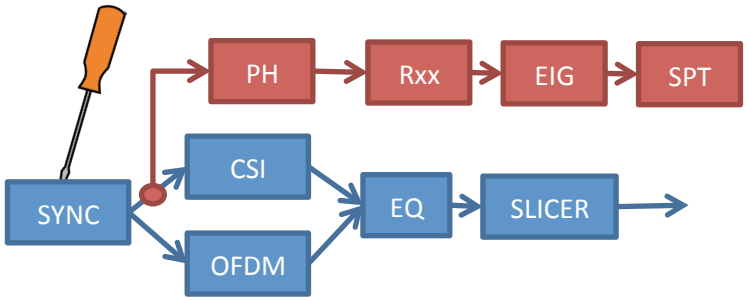


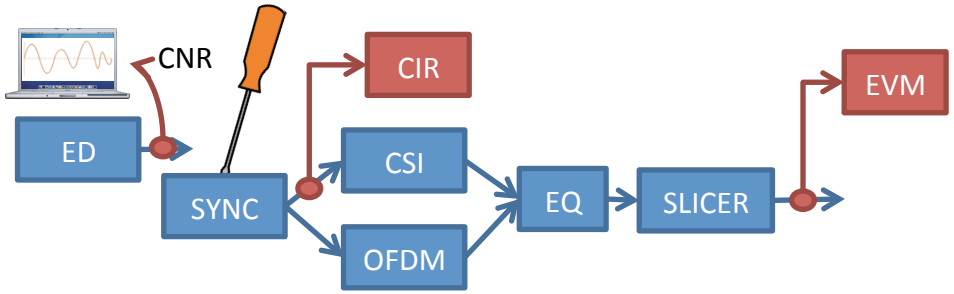
Base-station software can be built with atoms

Atomix WiFi chain can meet throughput and latency

Apps are easily added to Atomix WiFi

Location-signature app





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

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

No change in WiFi packet decode latency

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: a new programming framework

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

Future work:

- Automated resource scheduling
- Static program checking
- Extending to L2-L7 NFV packet processing