# MCHEM: At-Scale Realistic Testing in a Channel Emulator

Ashish Chaudhari

Principal Software Architect

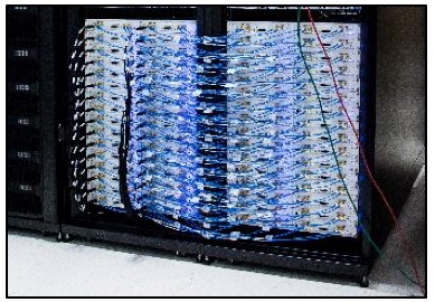
# Where This Talk Fits in the Overall Workshop

DARPA Spectrum Collaboration Challenge – Challenges

## Collaborate Without Co-Design

*Create radio networks that work with others without knowing how they "think"*

## Engineer Emergent Effects



*Discover and solve issues that only arise in large-scale realistic settings*

## Communicate Without Constraints

***too specific***

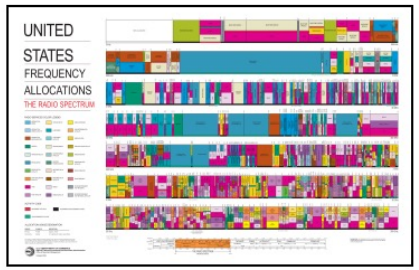
*frame 15, slot 7*

***too general***

*A B C → D*

*Create a protocol that supports evolving new forms of collaboration*

## Evolve The Ecosystem




*Change radio design, applications, and spectrum management to enable and leverage collaboration.*

Thumbs-up image source: http://sr.photos3.fotosearch.com/bthumb/CSP/CSP880/k8803233.jpg
Pencil image source: http://www.pngall.com/wp-content/uploads/2016/03/Pencil-PNG.png



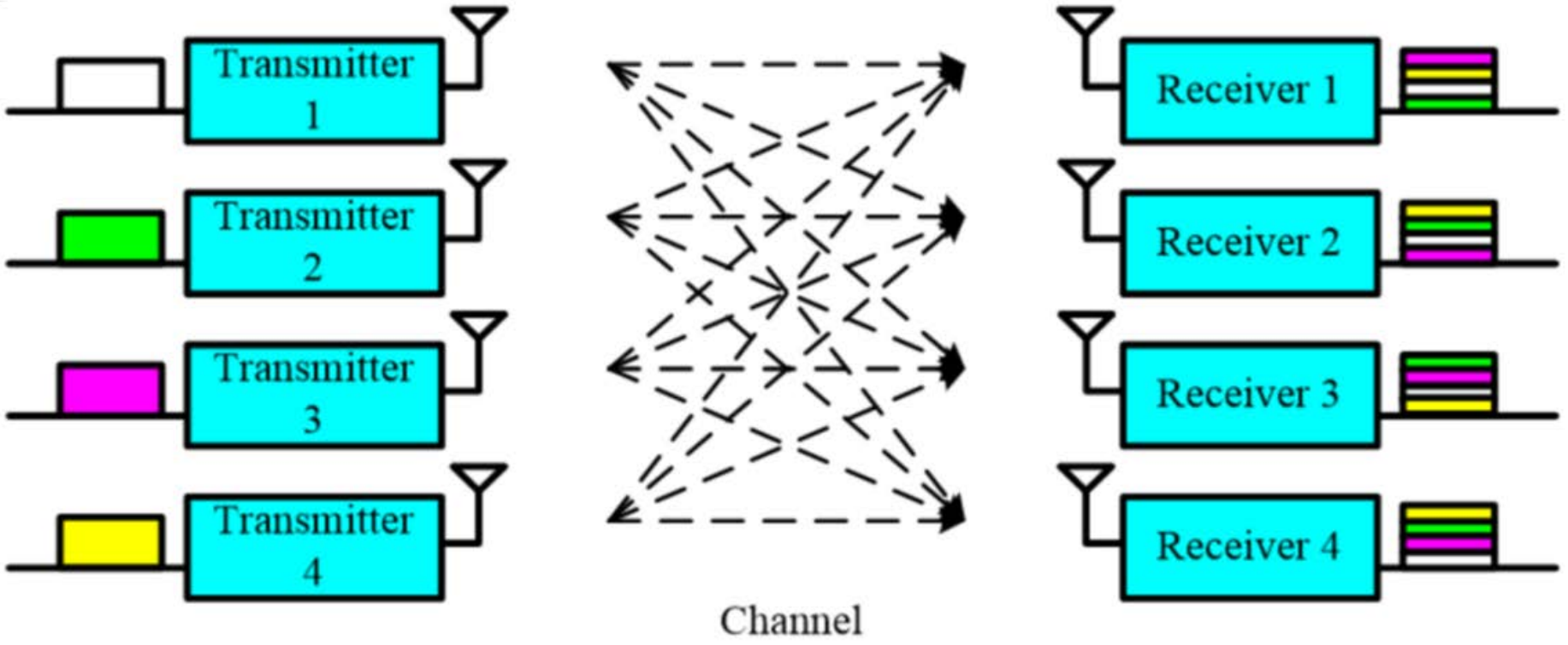
NATIONAL INSTRUMENTS

# Agenda

- Background: Channel Emulators
  - What is a channel emulator?
  - Why use a channel emulator?
- MCHEM Data Plane Architecture
  - Radio and Processing Nodes
  - Sample Topology
- MCHEM Control Plane Architecture
  - Software API
  - Protocols
  - Control Connectivity

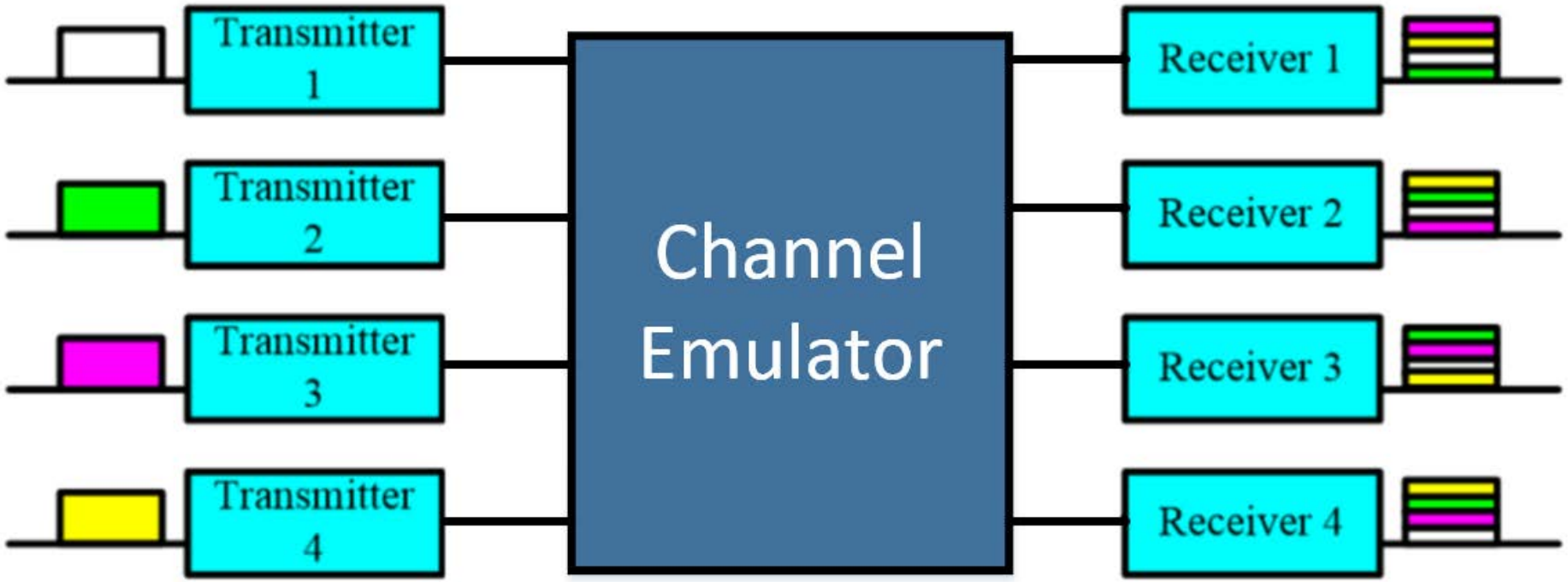
# MCHEM is a Massive Channel Emulator

*In a test environment, channel emulators replace the real-world radio channel between a radio transmitter and a receiver by providing a faded representation of a transmitted signal to the receiver inputs.*



# MCHEM is a Massive Channel Emulator

*In a test environment, channel emulators replace the real-world radio channel between a radio transmitter and a receiver by providing a faded representation of a transmitted signal to the receiver inputs.*

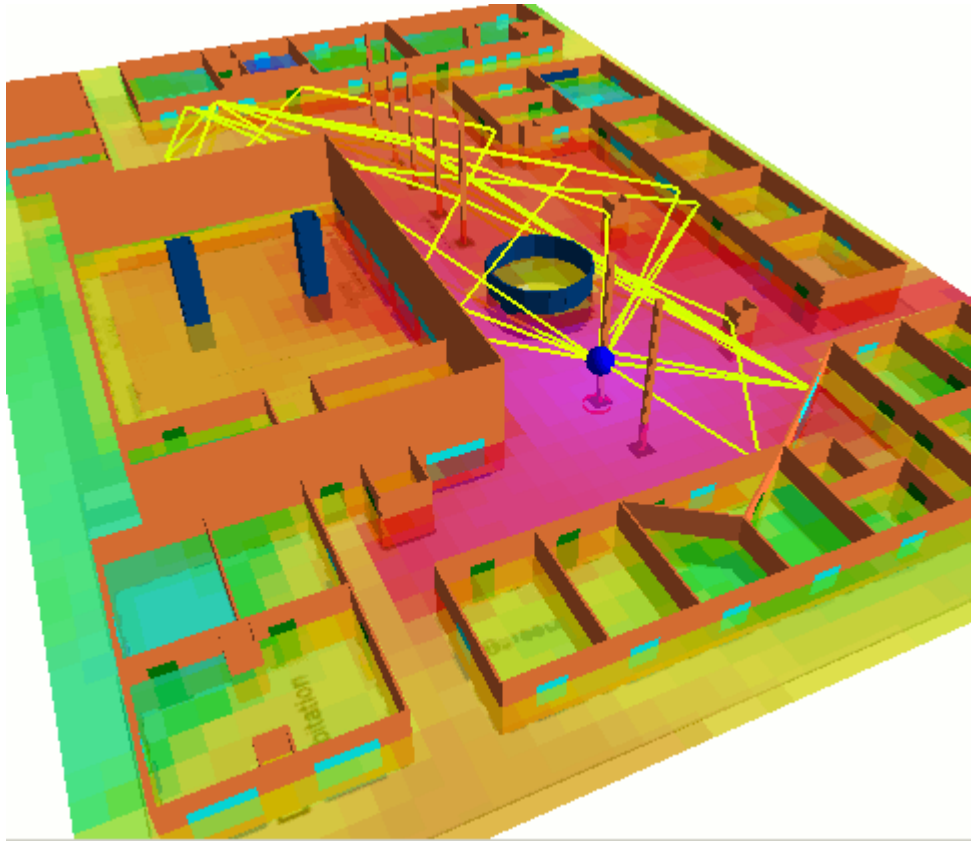


# Channel Emulator Uses

To model real-world RF environments

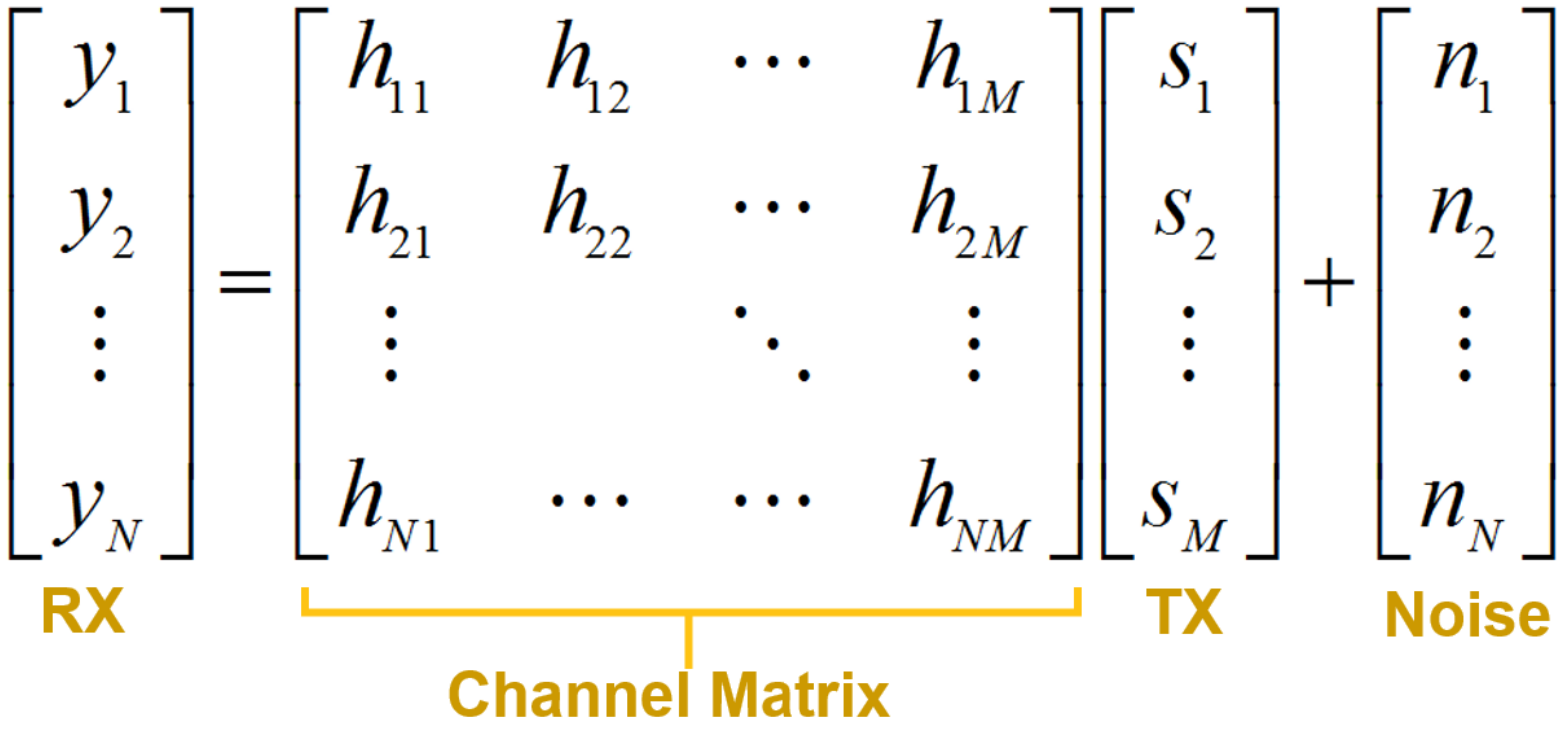
- Accurately
- Deterministically
- Flexibly

Model an environment and a time varying scenario without actually building it.



# Channel Emulator Computation

- In an N-channel emulator, the contribution of every channel on every other channel is computed
- Matrix-multiplication but with filters
- Computational complexity scales with $N^2$



# MCHEM Data Plane Architecture

# MCHEM Hardware Capabilities

- Antennas: 256 (in a full mesh 256x256 configuration)
- Bandwidth: 80MHz (100MS/s)
- Filter: 4-tap, 5us delay line (500 samples)
- Channel Update Rate: 1kHz
- Latency: ~15us

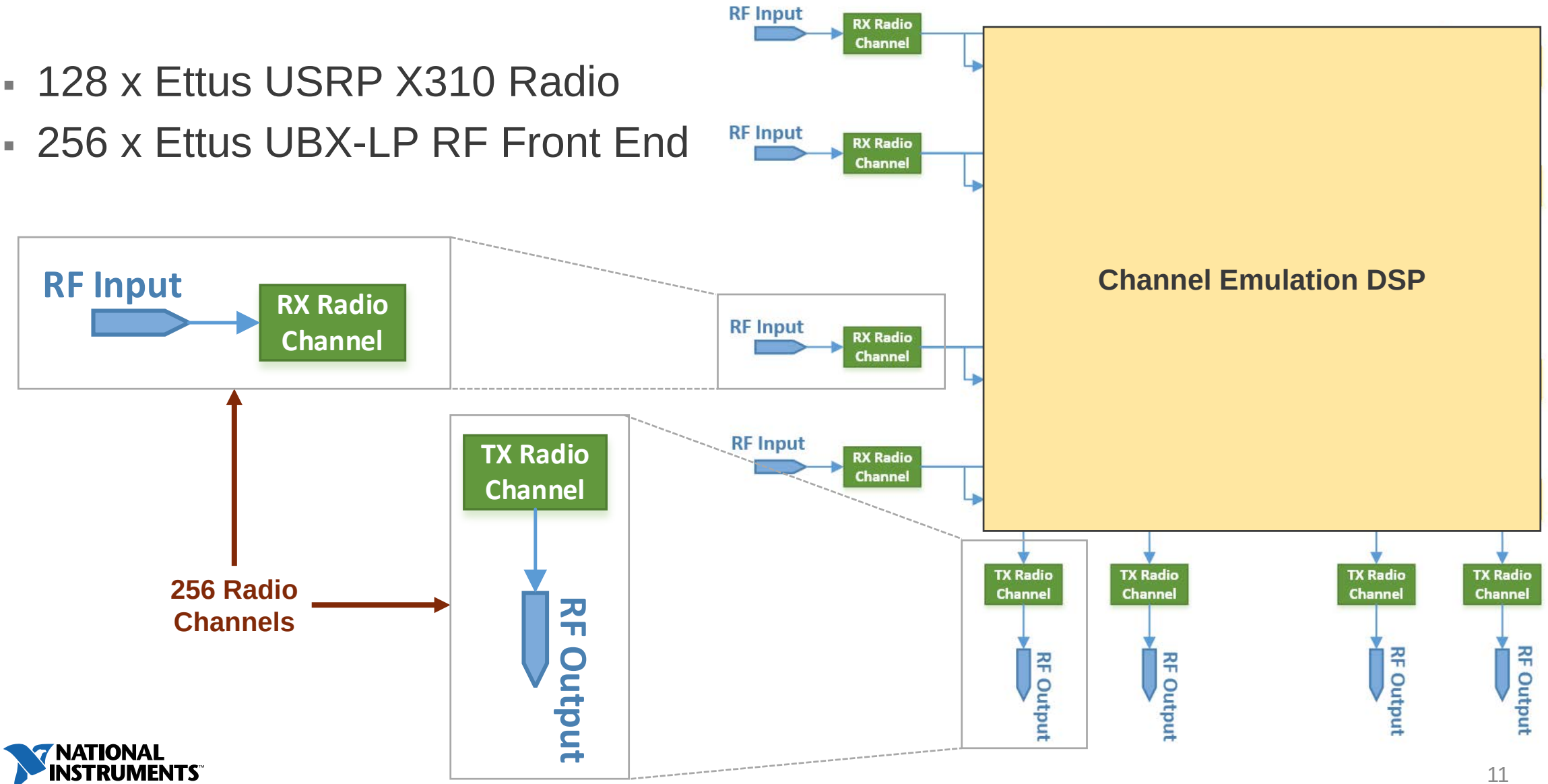
By Diliff (Own work) [CC BY-SA 2.5 (https://creativecommons.org/licenses/by-sa/2.5)], via Wikimedia Commons

# MCHEM Block Diagram (Radios)

- 128 x Ettus USRP X310 Radio
- 256 x Ettus UBX-LP RF Front End

# MCHEM Block Diagram (Radios)

- 128 x Ettus USRP X310 Radio
- 256 x Ettus UBX-LP RF Front End



# MCHEM Block Diagram (DSP)

- 256x256 Sparse Finite Impulse Response (FIR) Filters
- 4 non-zero complex taps arbitrarily placed in a 500 sample delay line

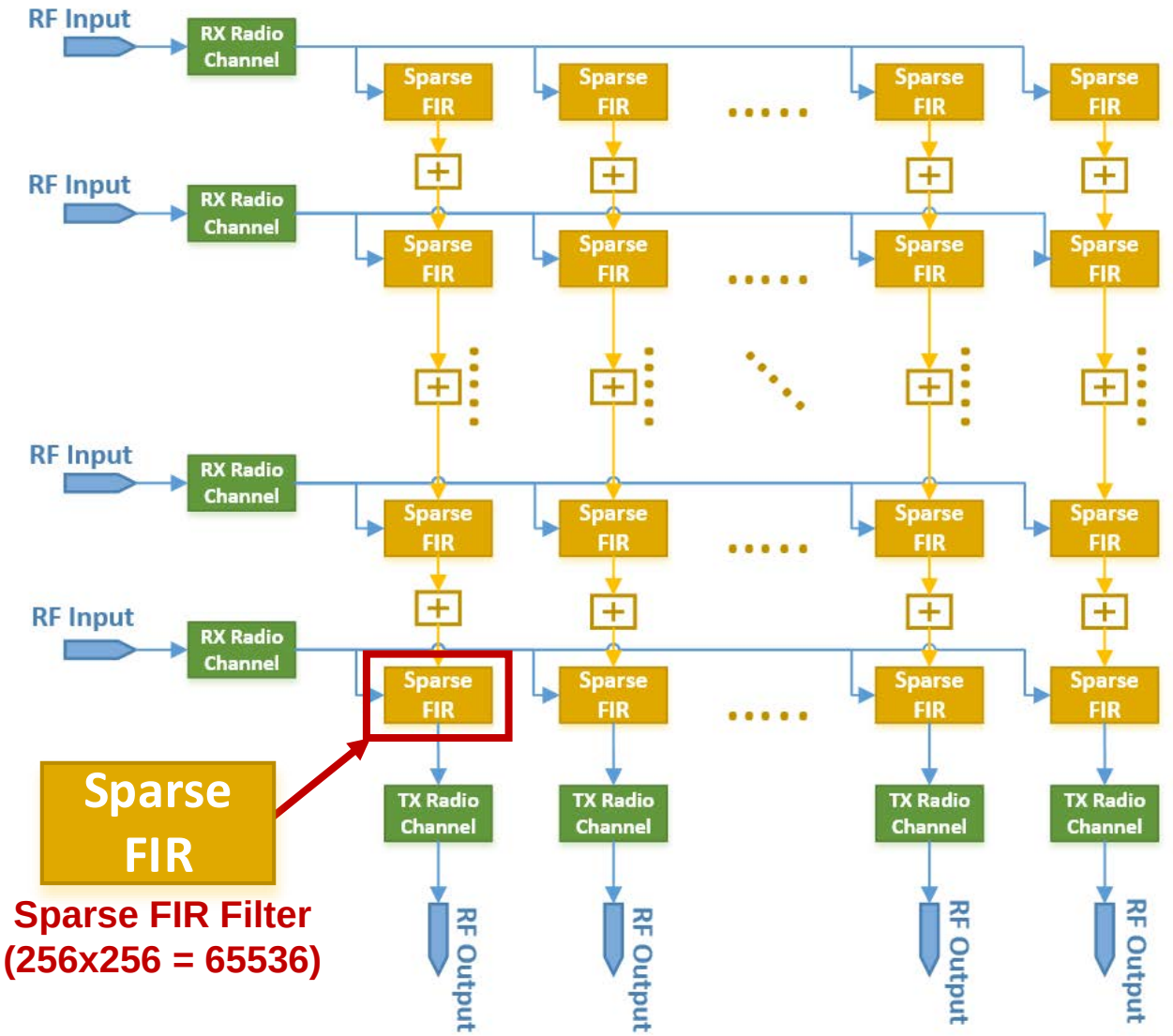
$$h[n] = \sum_{i=1}^{N} c_i \, \delta[n - k_i]$$

*where:*

*N = Number of coefficients (Target = 4)*

*$c_i$ = Coefficient (complex: 16 bit I and 16 bit Q)*

*$k_i$ = Sample delay value (integer: 0 - 511)*

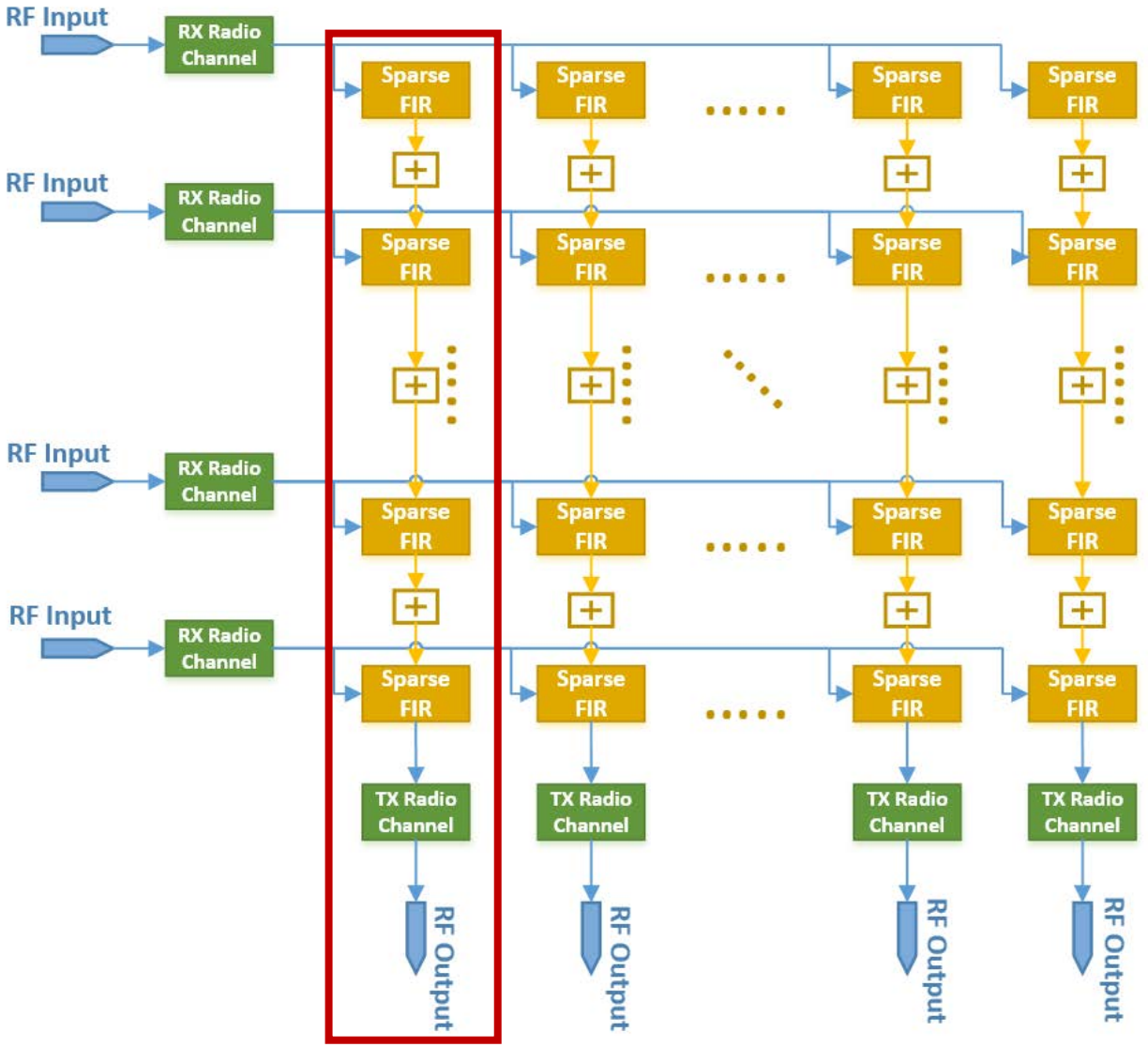


# Sample Bandwidth Requirements

Input Bandwidth to Compute **ONE** Output Channel

= (4 Bytes/Samp) x
(100 MSamp/s per chan) x
(256 chan)

= **~102 GigaBytes/s**
(100 Gen3 PCIe lanes)

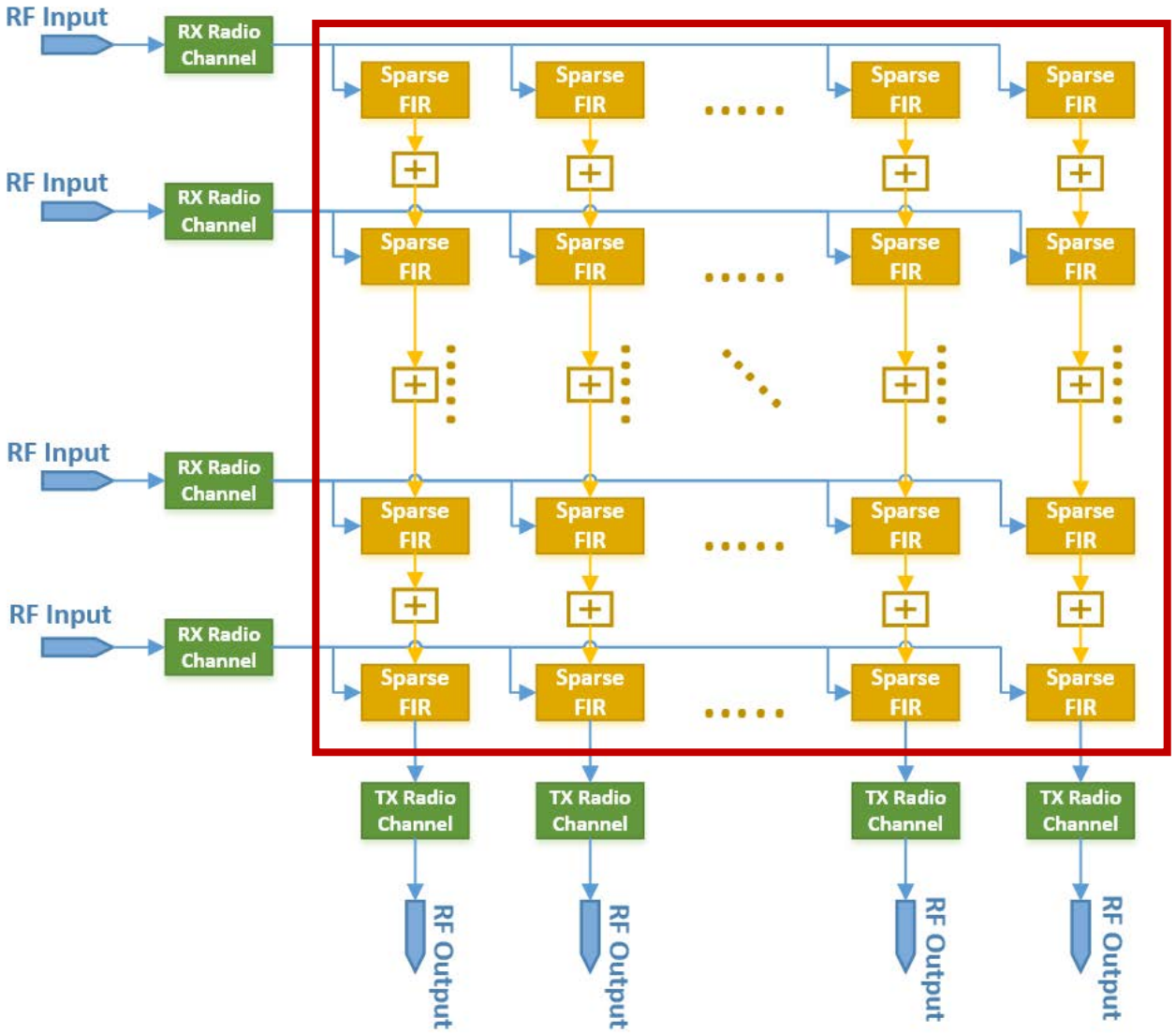


# Computation Throughput Requirements

DSP Bandwidth to Compute 256x256 Channels

= (3 DSP Ops/tap) x
(4 taps/filter) x
(65536 filters) x
(100MS/s)

= **~78 TeraMAC/s**
(Multiply-Accumulates)



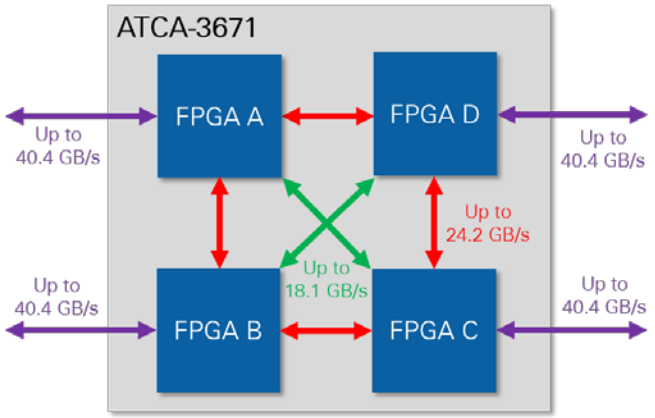
# FPGA Based Computation

Software Implementation is Impractical

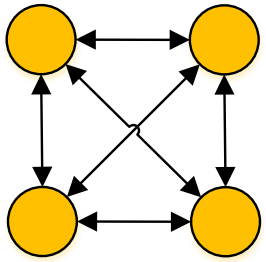
- High Data Movement Bandwidth Requirement
- High Compute Throughput Requirement
- Data Aggregation Bottlenecks

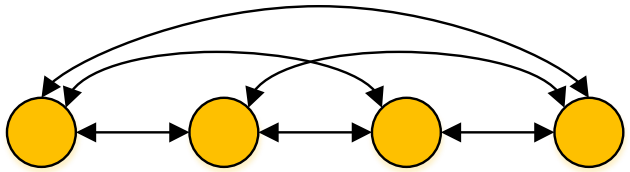
MCHEM uses the NI ATCA-3671 Accelerator

- Four Virtex7-690T FPGAs
- High Connectivity between FPGAs
  - Internal connectivity over PCB
  - External connectivity using RTM and Front-Panel IO



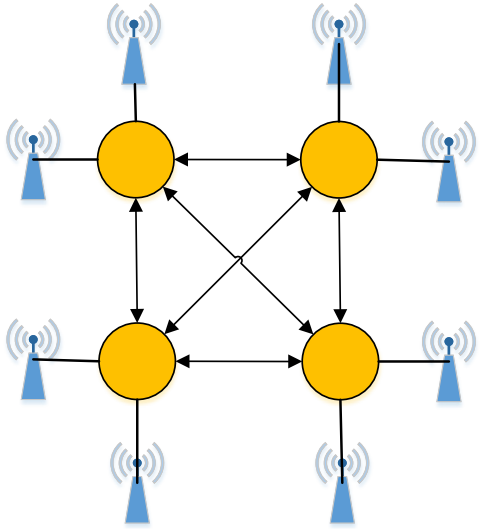
# Sample Movement Topology

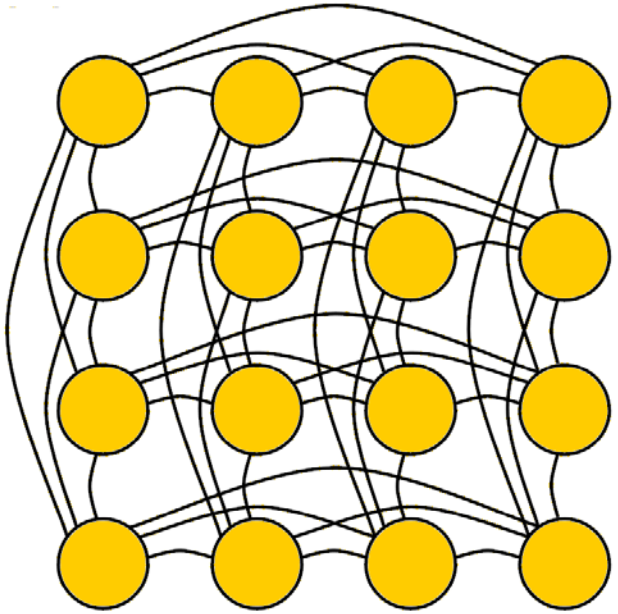




**Four Virtex7 FPGAs**

**Four Virtex7 FPGAs**
**+**
**Eight USRP X310s**

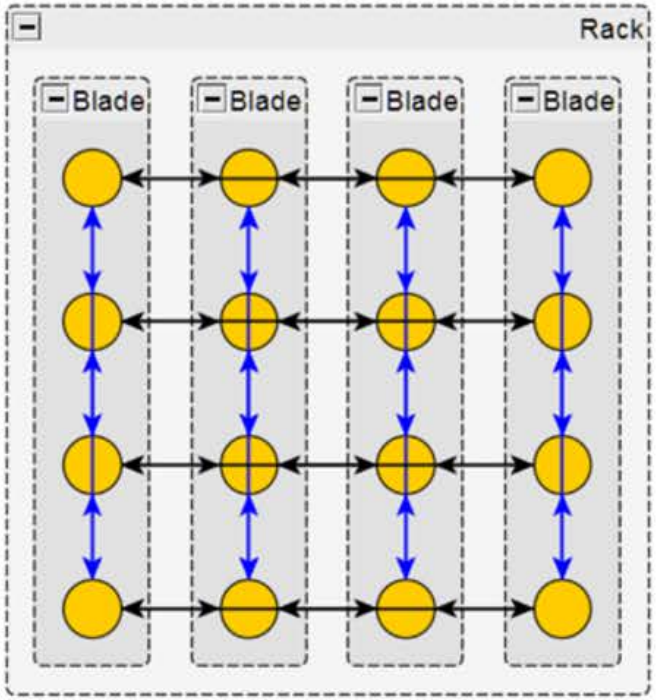




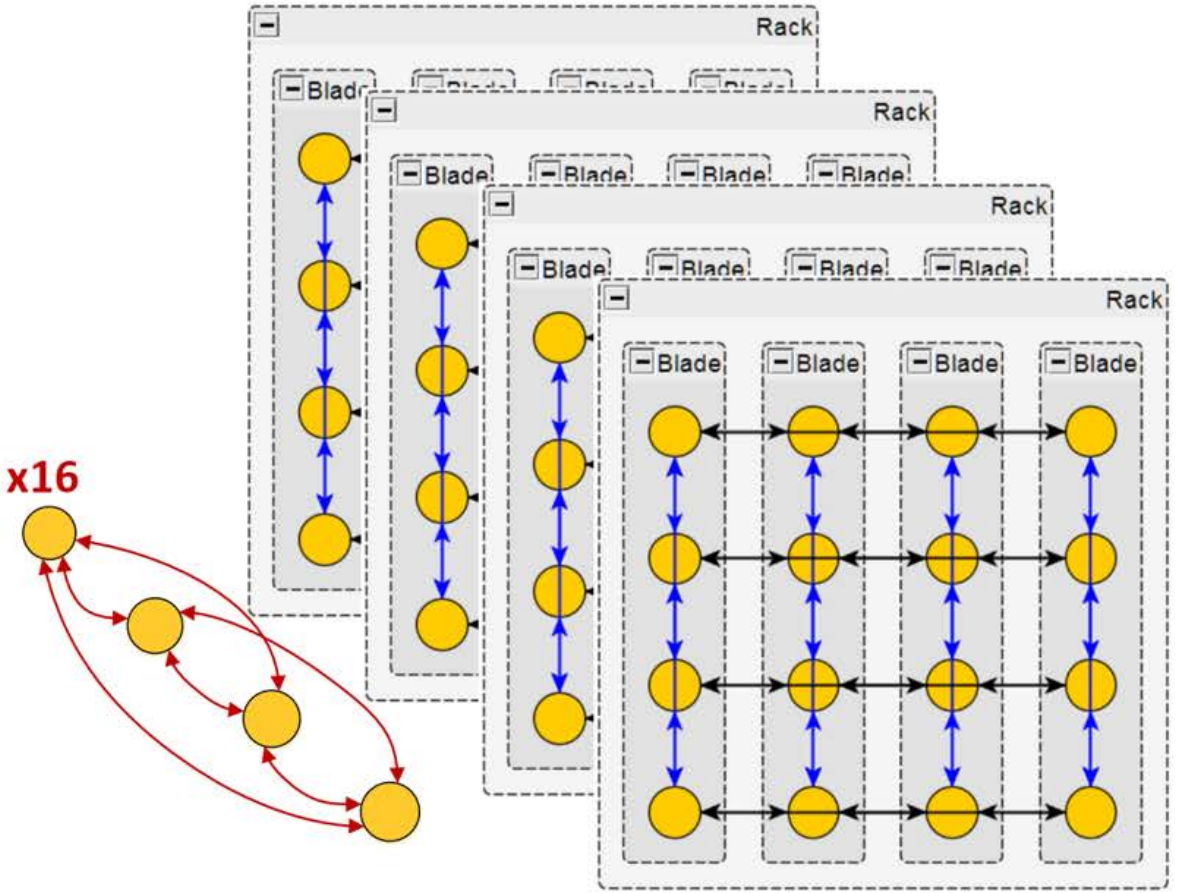
**Sixteen Virtex7 FPGAs**
**(Full Dim Connectivity)**

**Quarter System**

# Sample Movement Topology (4x4x4 HyperX[1])




A Quarter Emulator




The Full Emulator

1. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.158.3130&rep=rep1&type=pdf

NATIONAL INSTRUMENTS

# MCHEM Control Plane Architecture

# MCHEM Software API

- Initialize Channel Emulator
- Start/Stop Dataflow
- Configure Radios (Freq., Gain, Dynamic Range, etc.)
- Configure Filters (Rate, Gain)
- Download Channel Taps
- Query Radio/Accelerator/Filter Status

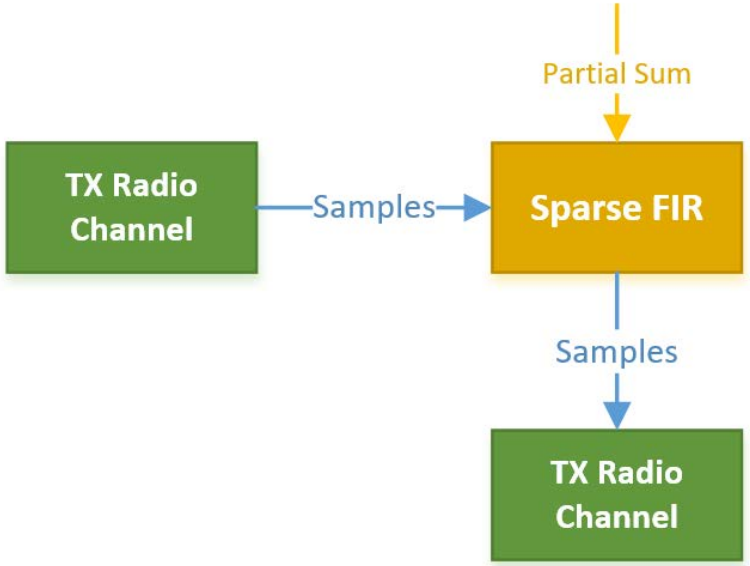
- Commands Issued over IP+UDP Link




Image Credits: https://www.xda-developers.com/why-you-should-prioritize-your-api-strategy-before-its-design/

# MCHEM Data Plane Revisited

- Samples to Accelerator (10G Aurora[1])
  - USRP FPGA ⇔ ATCA FPGA
- Samples from Accelerator (10G Aurora[1])
  - ATCA FPGA ⇔ USRP FPGA
- Partial Sums between Accelerators (10G Aurora[1])
  - ATCA FPGA ⇔ ATCA FPGA

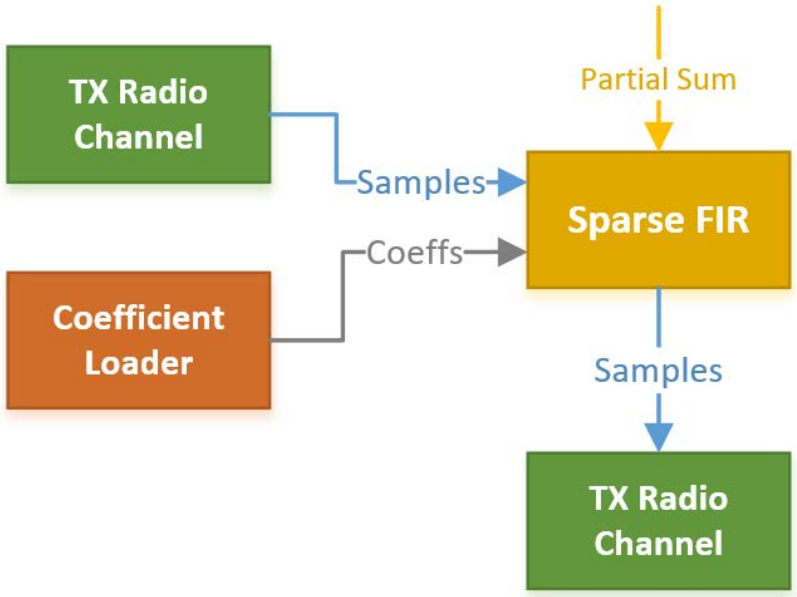



- Packetized Transport (RFNoC CHDR[2])
- Packet Buffer Flow Control
- Lossy with CRC Validation

1. https://www.xilinx.com/products/intellectual-property/aurora64b66b.html
2. https://kb.ettus.com/Getting_Started_with_RFNoC_Development

# MCHEM Coefficient Plane

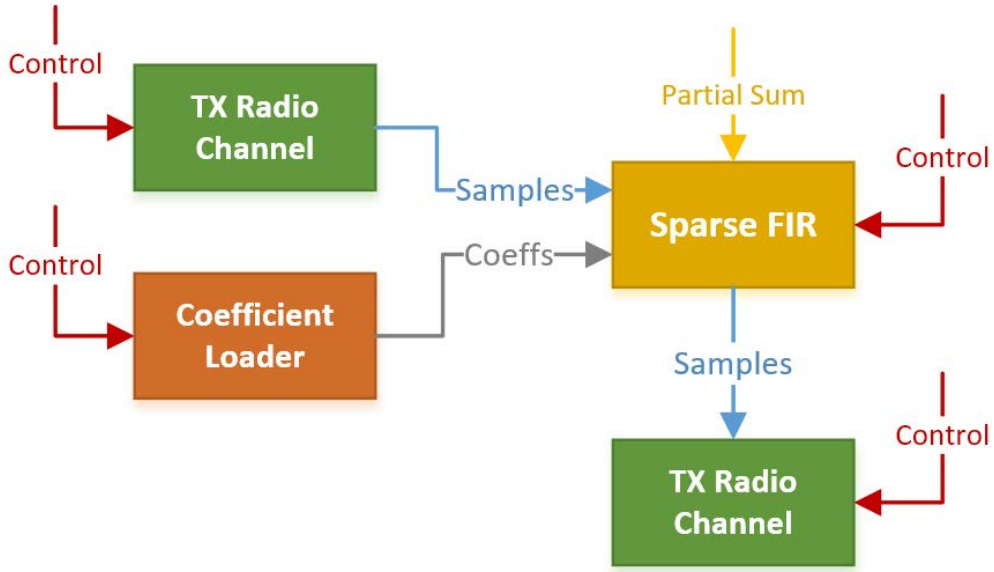
- Coefficient Download (10G Ethernet)
  - Server ⇔ ATCA FPGA




- Packetized Transport (IP + UDP + RFNoC CHDR)
- Packet Buffer Flow Control
- Lossy with CRC Validation

# MCHEM Control Plane

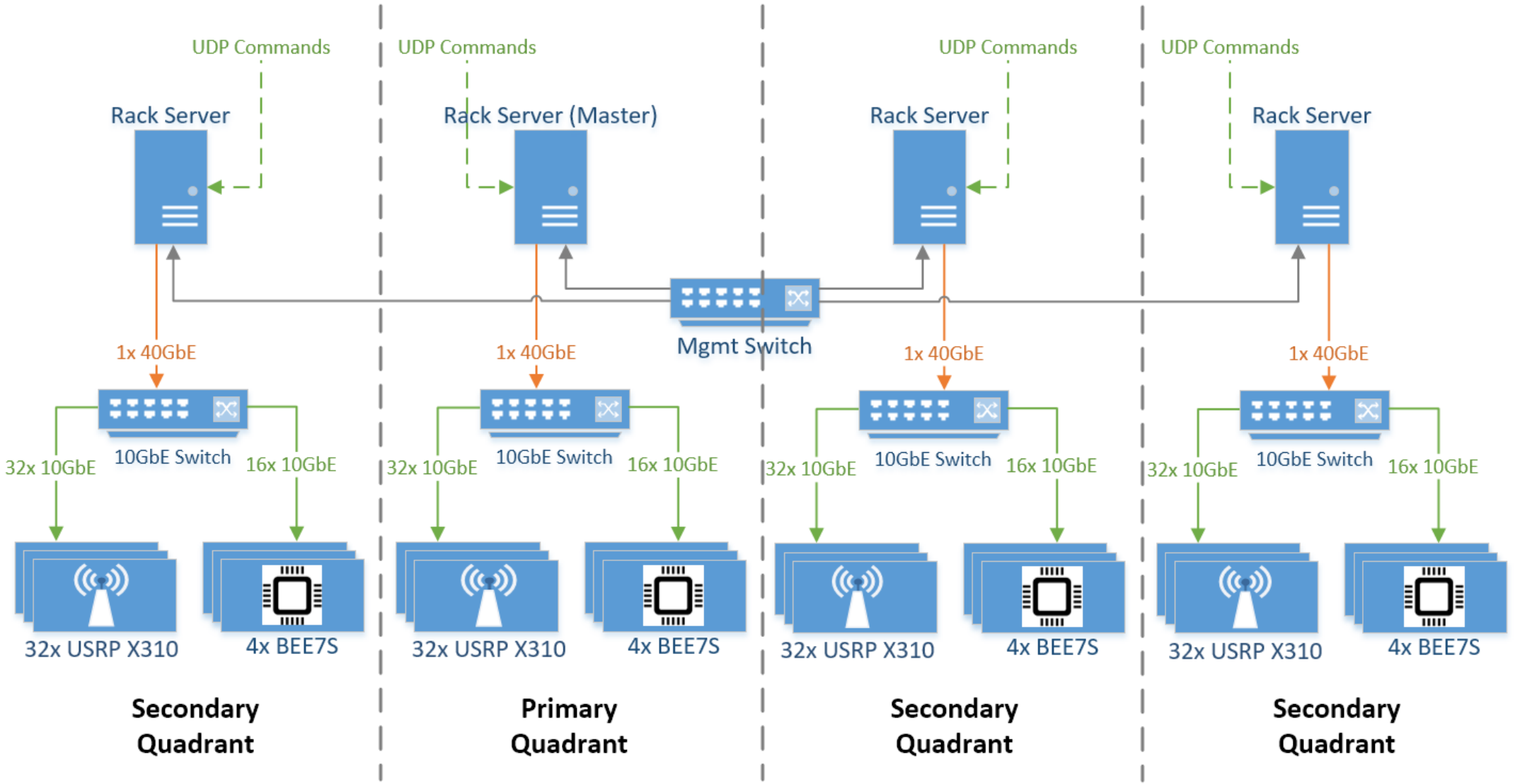
- Radio Control/Status (10G Ethernet: UDP)
  - Server ⇔ USRP
- Accelerator Control/Status (10G Ethernet: UDP)
  - Server ⇔ ATCA FPGA
- Server to Server (1G Ethernet TCP)
  - Server ⇔ Server




- Packetized Transport (Custom + Ettus UHD[1])
- Lossy with CRC Validation
- Application Level Retransmission

1. https://www.ettus.com/sdr-software/detail/usrp-hardware-driver

# Ethernet Connectivity



# Takeaways

# Conclusion

- The MCHEM allows emulating real-world, time-varying wireless environments in a very flexible manner
- The core of the emulator DSP is implemented in the FPGA which makes it low-latency, deterministic and real-time
- The scalable software API abstracts out control complexity and allows downloading arbitrary RF scenarios


NATIONAL
INSTRUMENTS™