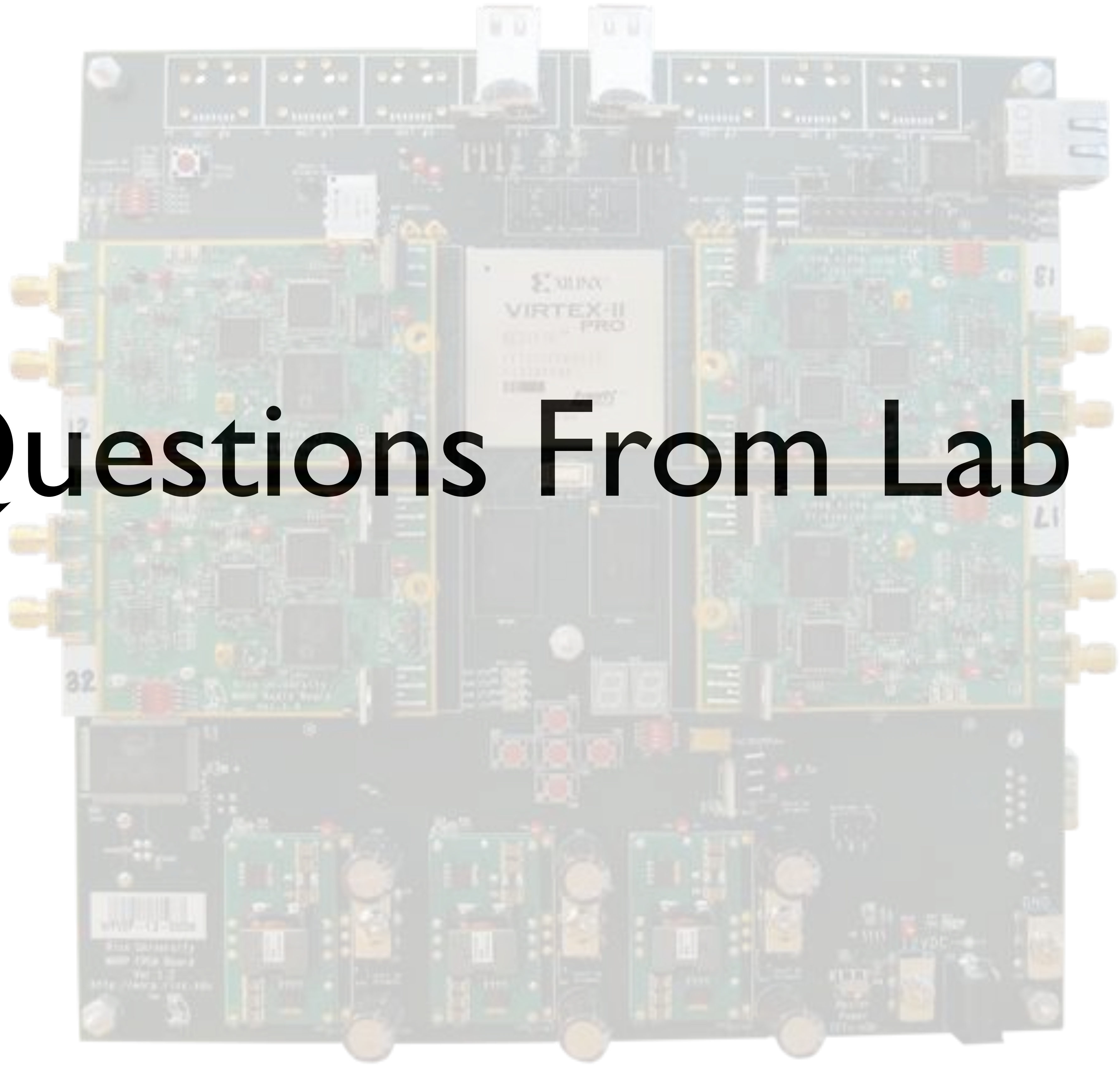
WARP: Physical Layer Design

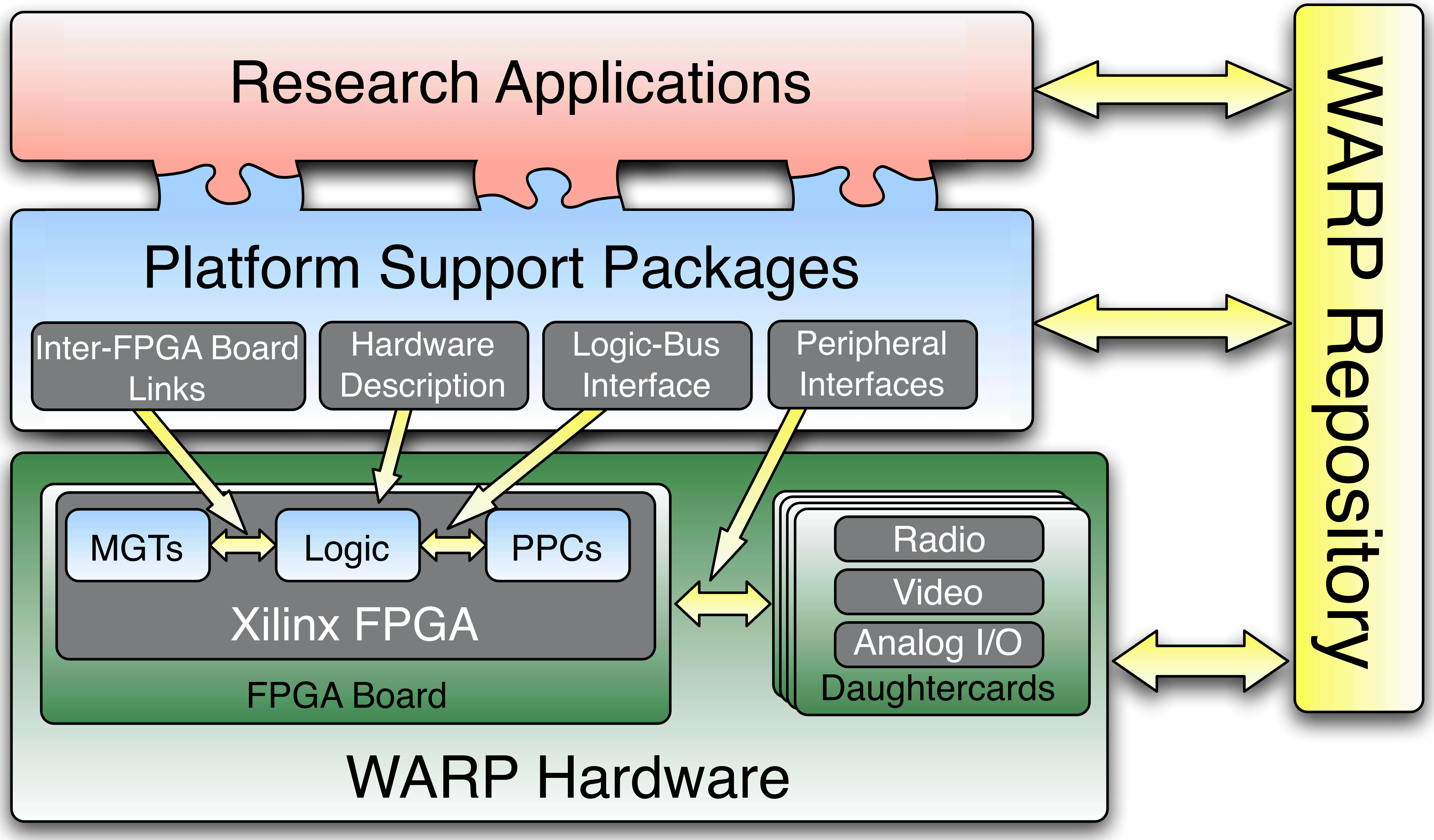
Patrick Murphy & Siddharth Gupta
Rice University

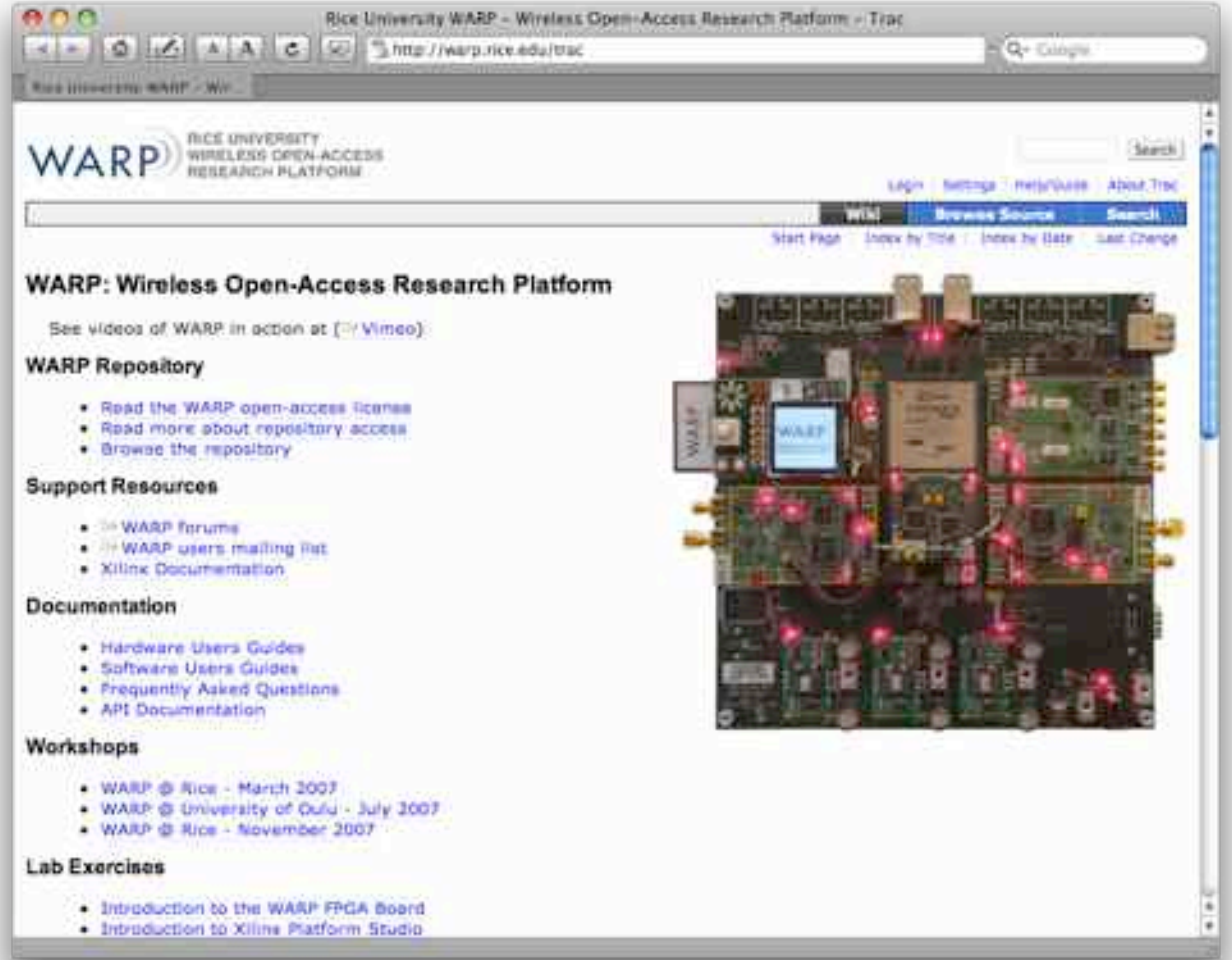
WARP Workshop
December 1, 2007

warp.rice.edu



Questions From Lab 1?


Research Applications
Platform Support Packages
Inter-FPGA Board Links
Hardware Description
Logic-Bus Interface
Peripheral Interfaces
MGTs
Logic
PPCs
Xilinx FPGA
FPGA Board
Radio
Video
Analog I/O
Daughtercards
WARP Hardware
WARP Repository



Rice University WARP – Wireless Open-Access Research Platform – Trac

http://warp.rice.edu/trac

WARP RICE UNIVERSITY WIRELESS OPEN-ACCESS RESEARCH PLATFORM

Login | Settings | Help/Guide | About Trac

Wiki | Browse Source | Search

Start Page | Index by Title | Index by Date | Last Change

WARP: Wireless Open-Access Research Platform

See videos of WARP in action at (Vimeo)

WARP Repository

- Read the WARP open-access license
- Read more about repository access
- Browse the repository

Support Resources

- WARP forums
- WARP users mailing list
- Xilinx Documentation

Documentation

- Hardware Users Guides
- Software Users Guides
- Frequently Asked Questions
- API Documentation

Workshops

- WARP @ Rice - March 2007
- WARP @ University of Oulu - July 2007
- WARP @ Rice - November 2007

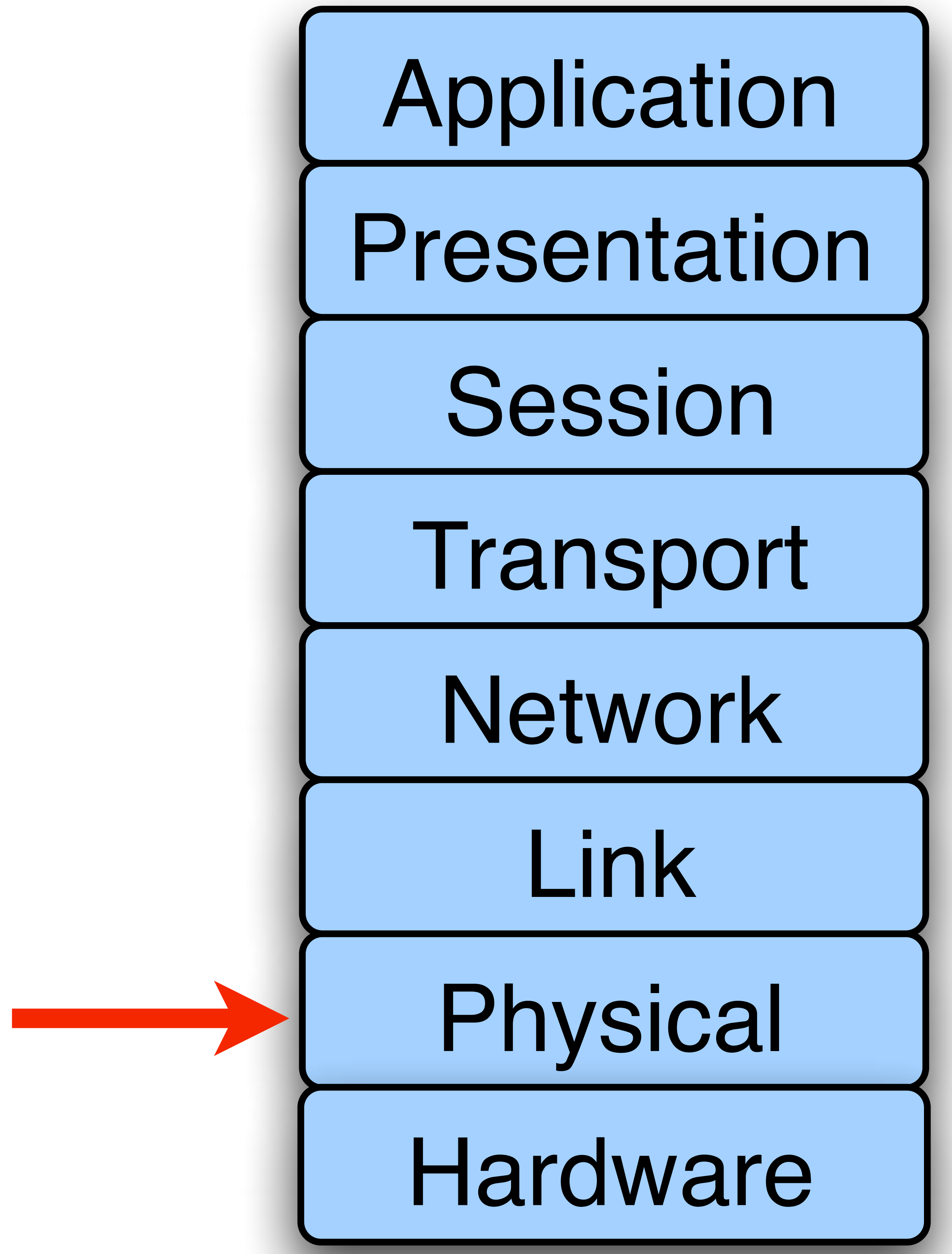
Lab Exercises

- Introduction to the WARP FPGA Board
- Introduction to Xilinx Platform Studio

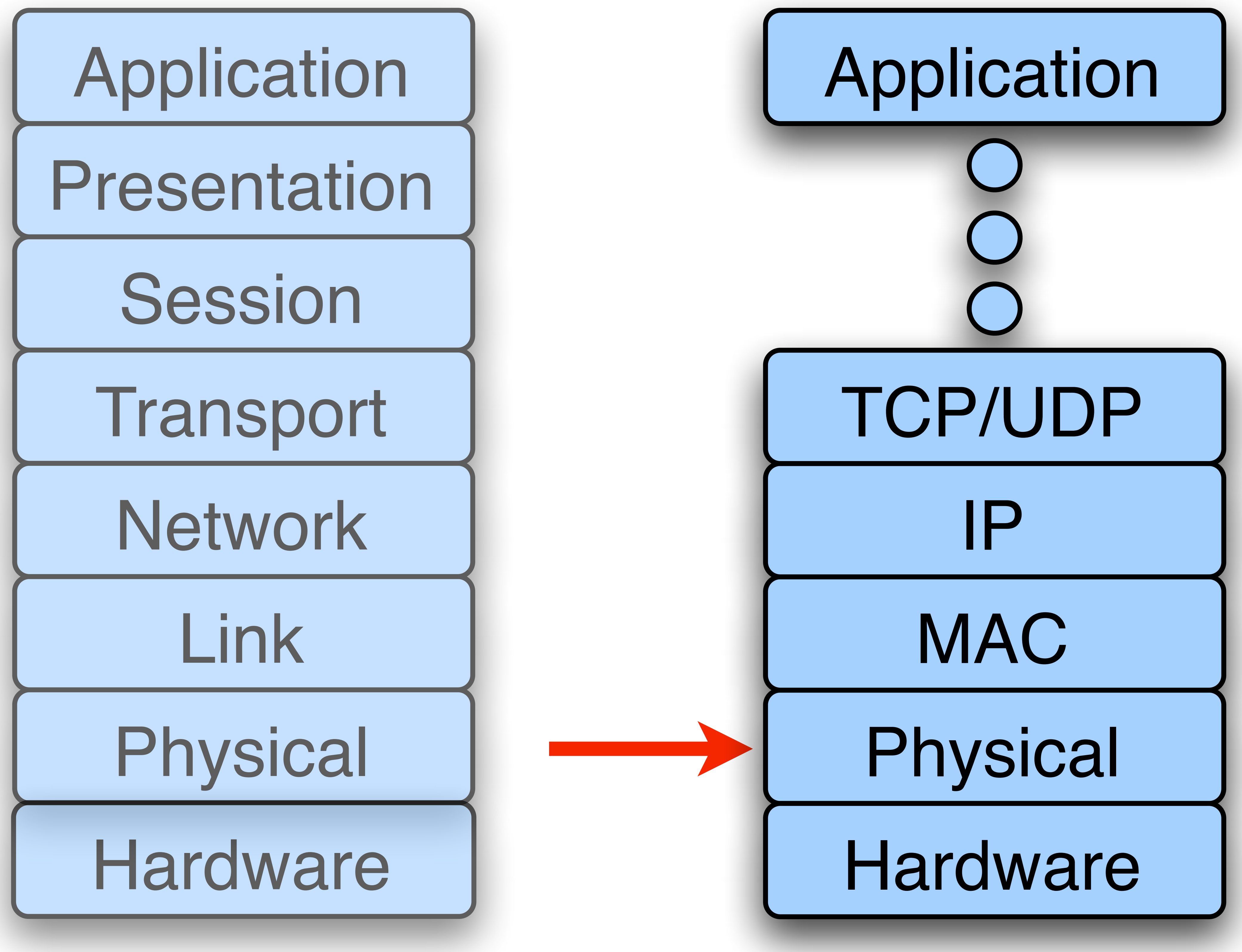
PHY Design - Outline

- Physical layer basics
- Real-time PHY design flow
- Introduction to WARPLab PHY design flow
- Lab 2: Building a simple transmitter
- Lab 3: Using WARPLab

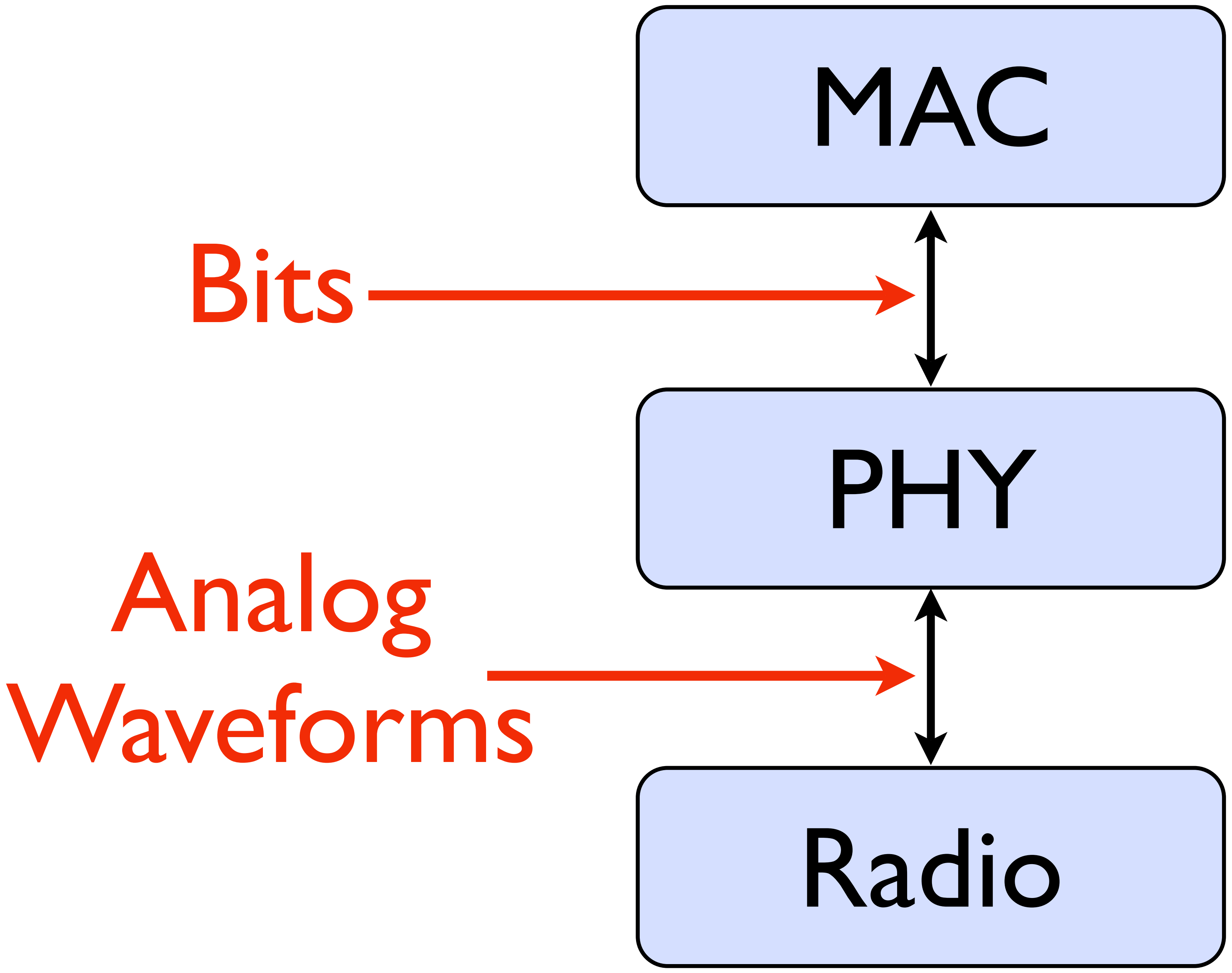
Physical Layer Basics



Physical Layer Basics

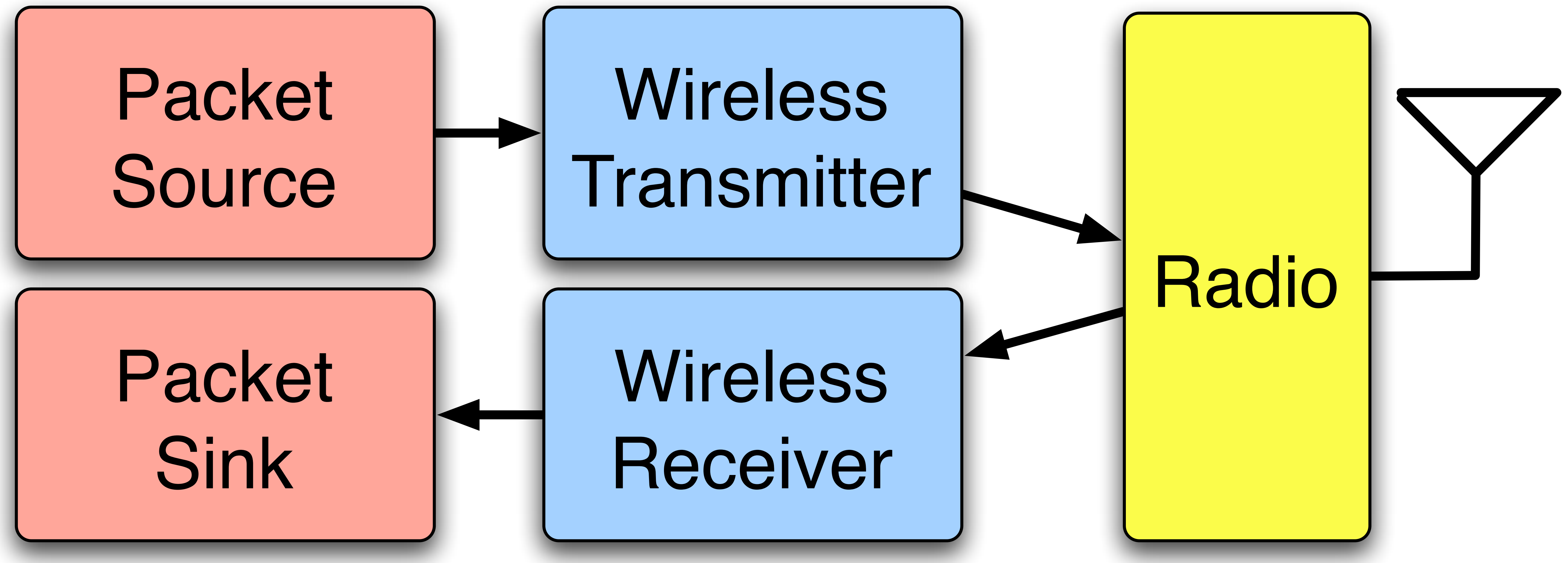


Physical Layer Basics



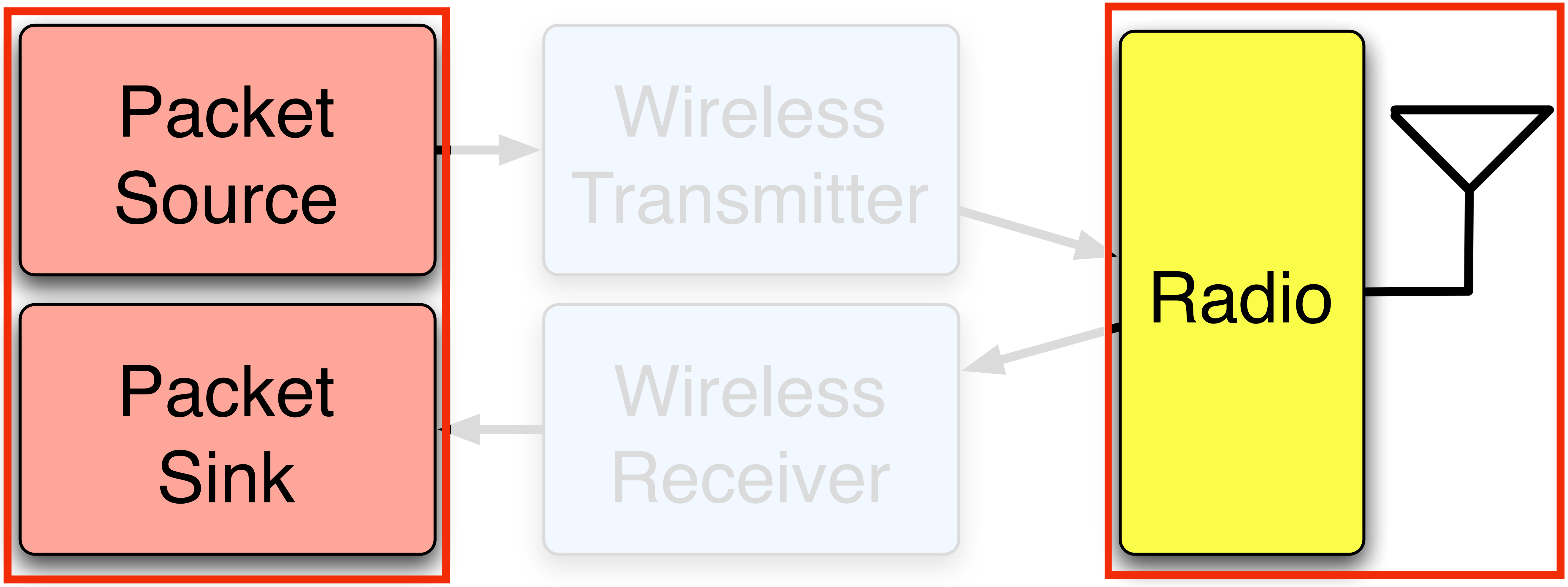
Physical Layer Basics

Simple Wireless Node



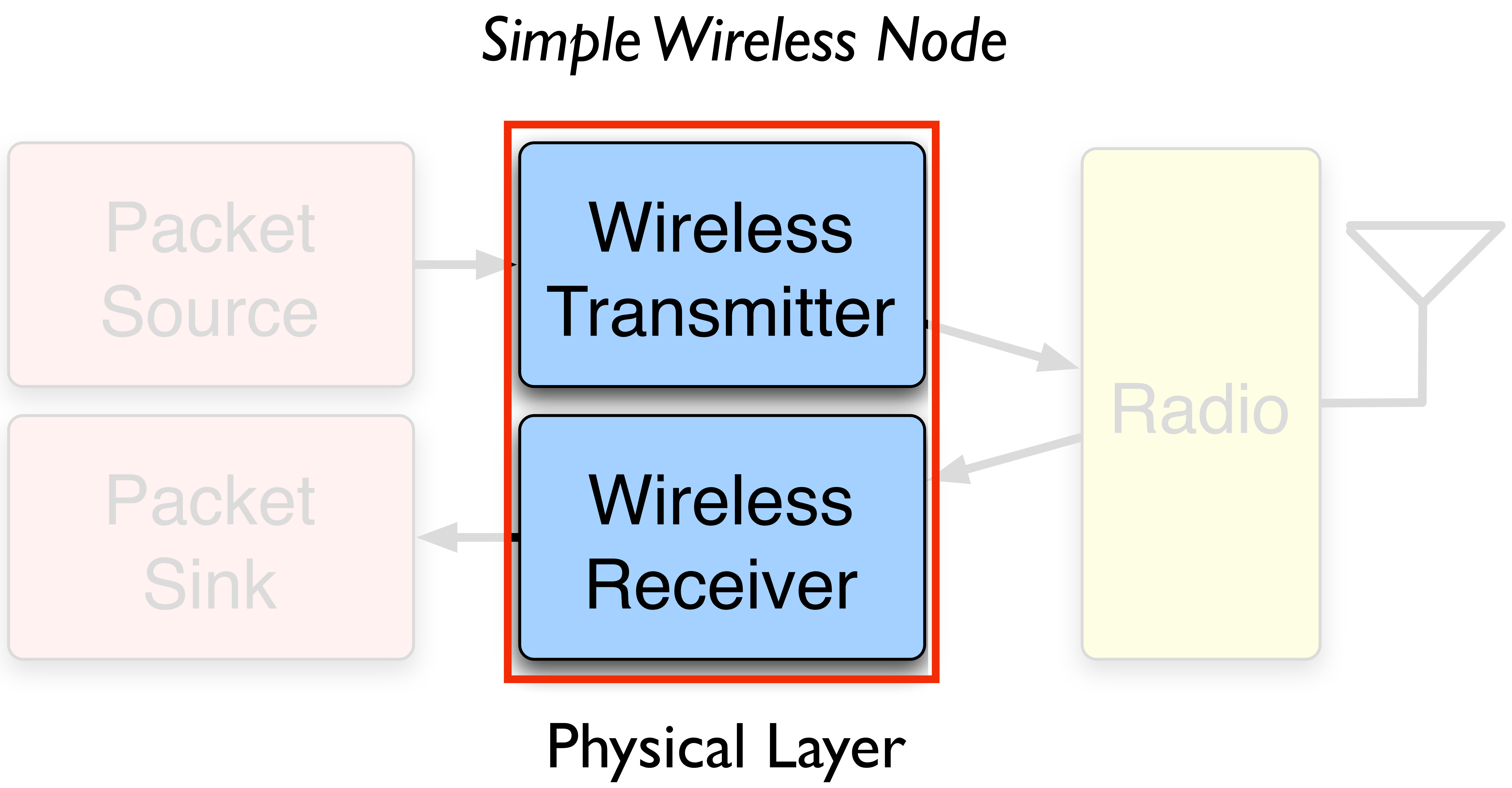
Physical Layer Basics

Simple Wireless Node



Somebody Else's Problem

Physical Layer Basics



PHY Design Flows

- Real-time PHY design
  - Low-level FPGA design
  - Putting it all together
- WARPLab
  - MATLAB↔WARP Link
  - Very rapid prototyping of PHY algorithms

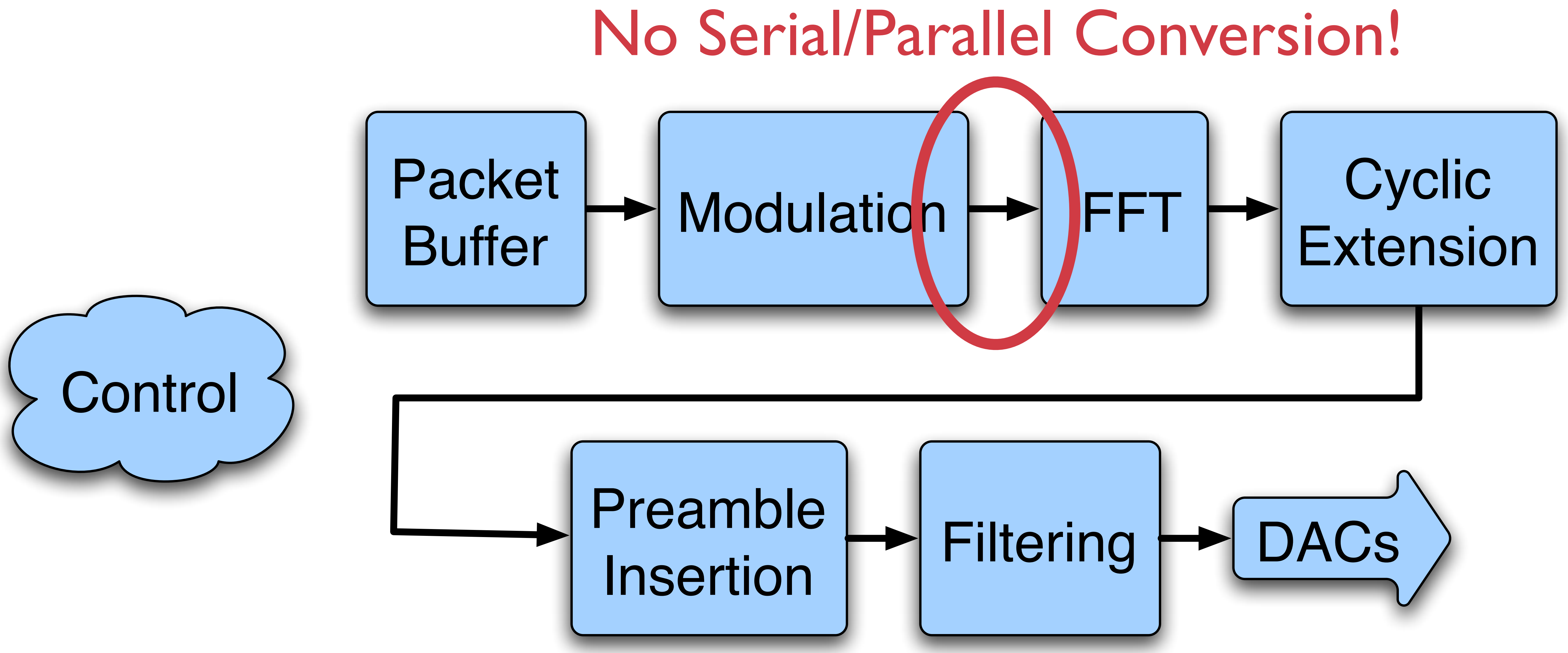
PHY Design Flows

- Real-time PHY design
  - Low-level FPGA design
  - Putting it all together
- WARPLab
  - MATLAB↔WARP Link
  - Very rapid prototyping of PHY algorithms

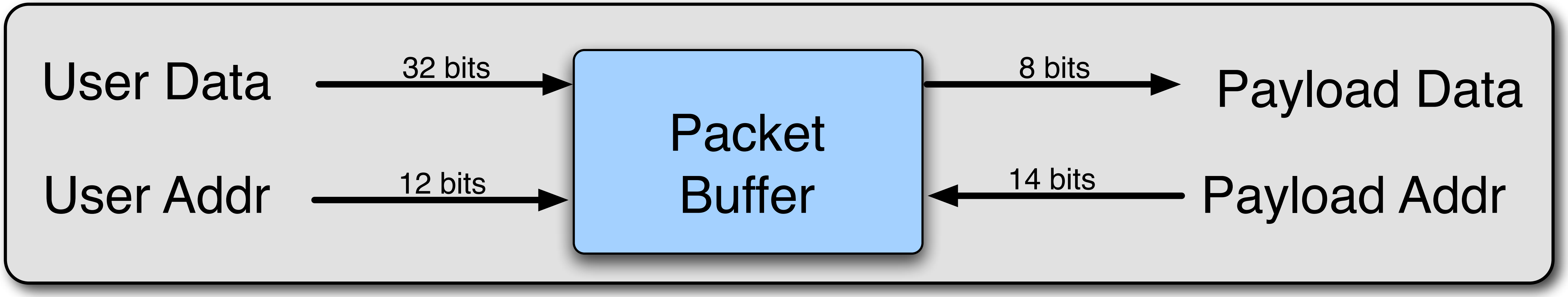
PHY Example: OFDM

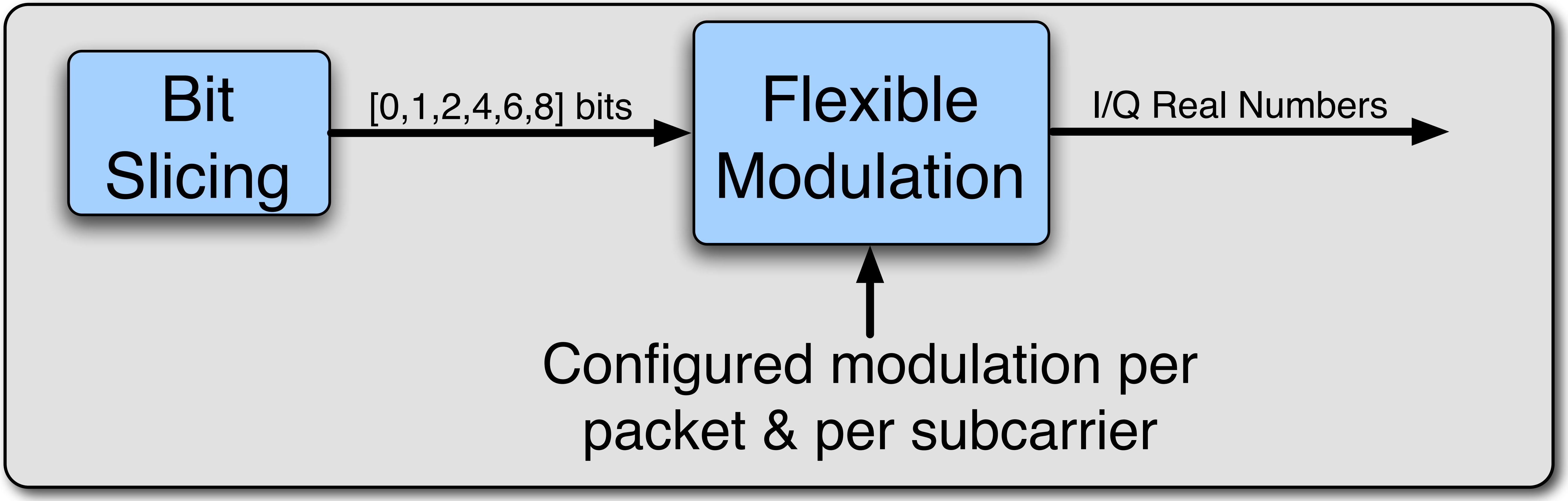
- Designed for wireless networking
  - Modeled on 802.11a (but not compliant)
- Packet-based OFDM transceiver
  - Packets source/sink in PowerPC code
- Wideband, real-time design
  - 4 cycles per sample
  - 10 MHz bandwidth at 40 MHz clock
- Full synchronization for standalone operation
- Implemented entirely in System Generator

PHY Example: OFDM Tx

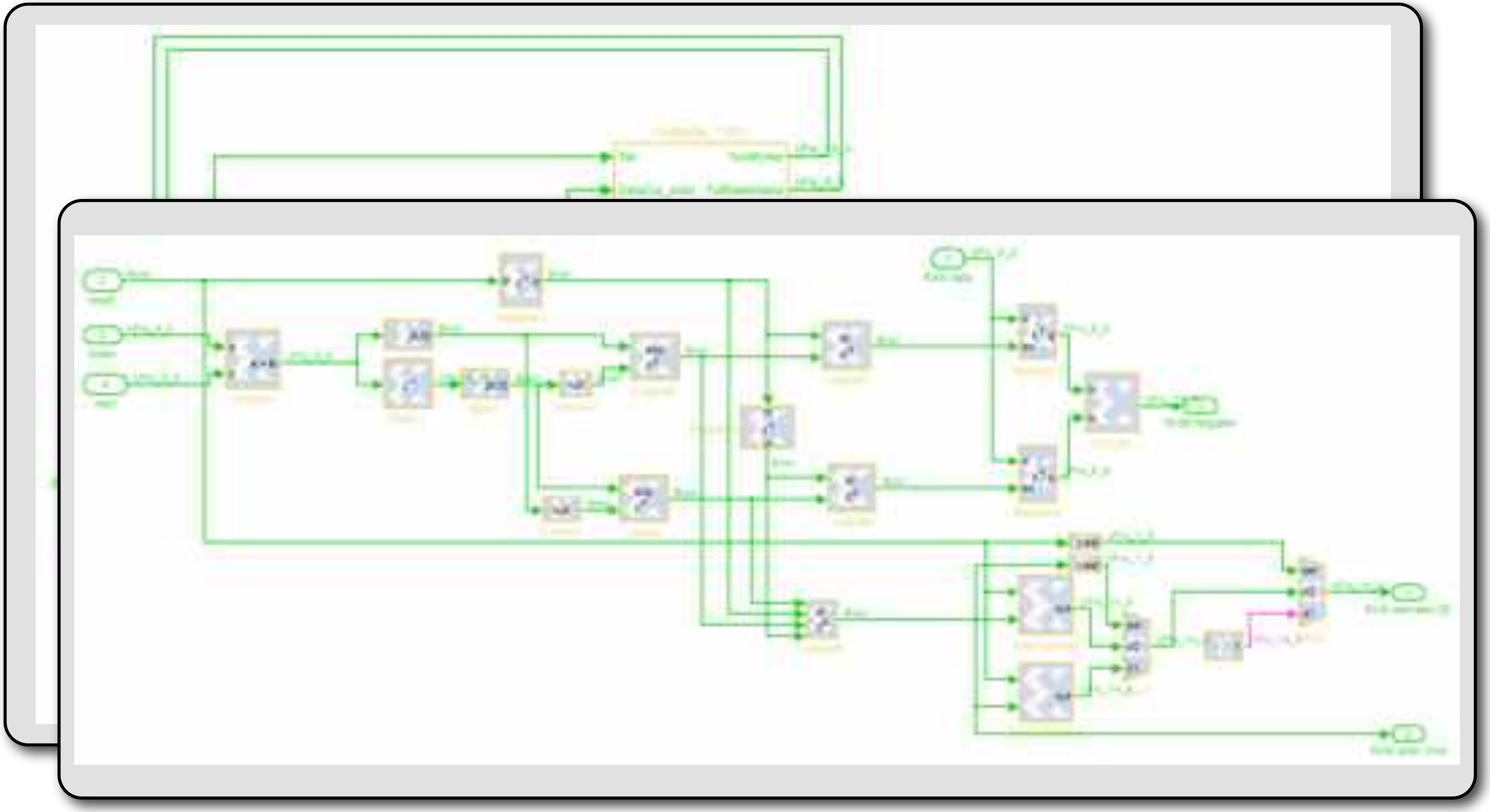


PHY Example: OFDM Tx



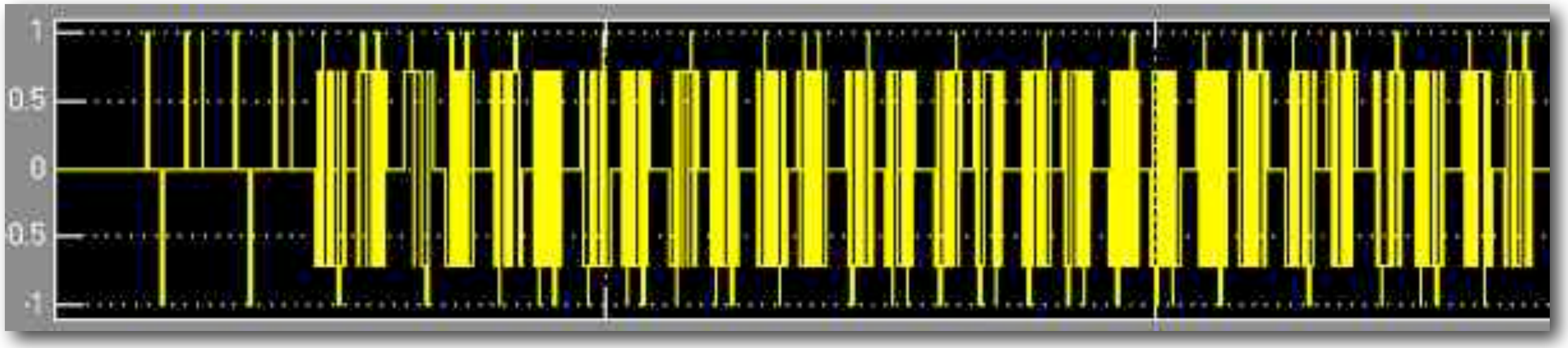


PHY Example: OFDM Tx

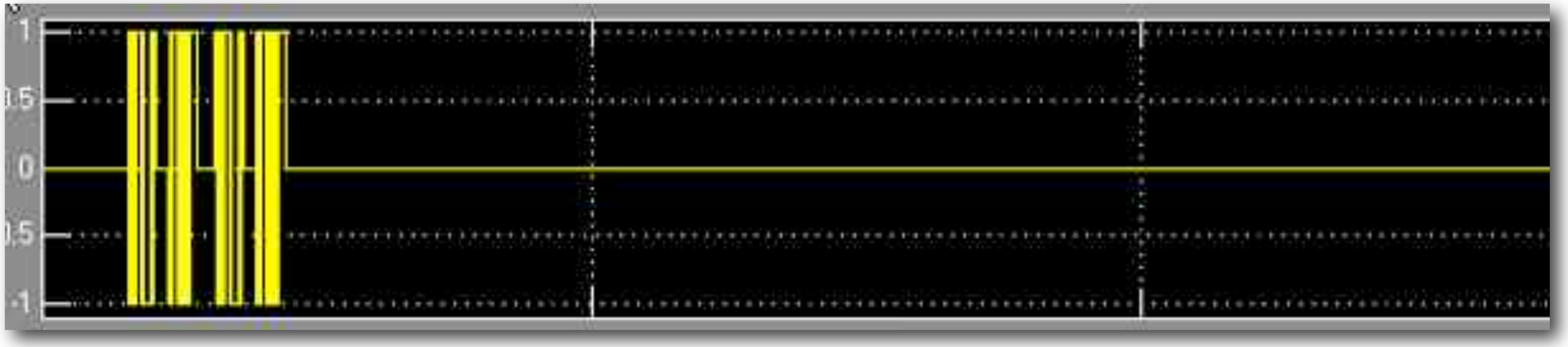


PHY Example: OFDM Tx

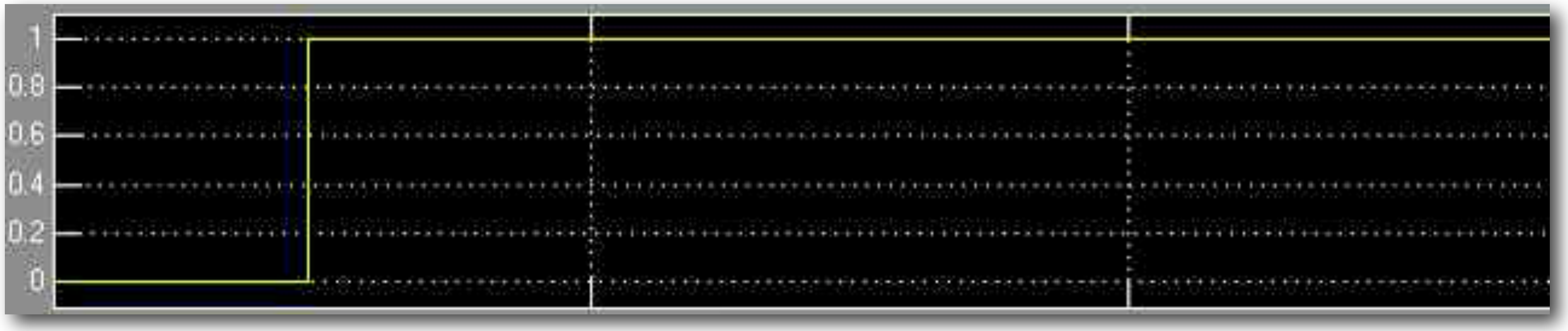
Modulator Output



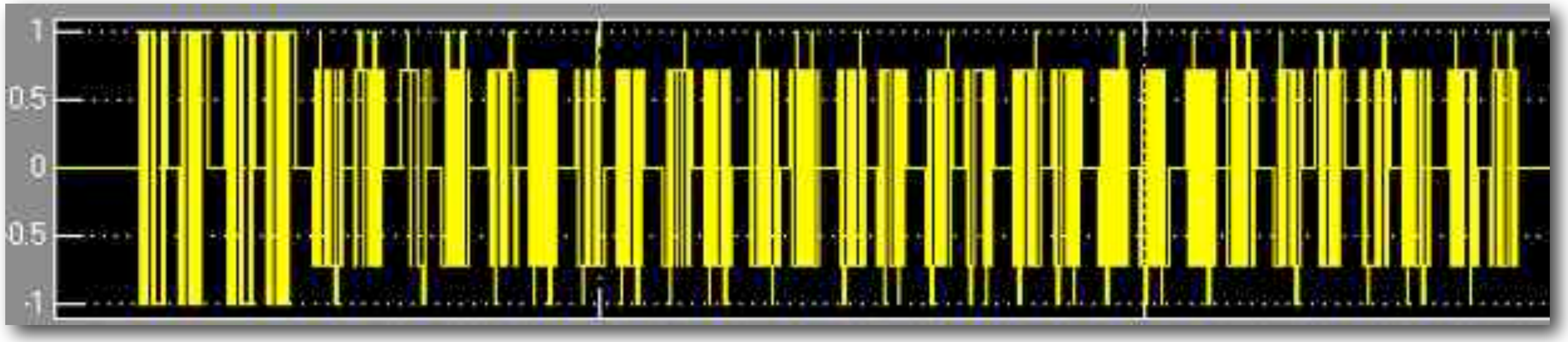
Stored Training Sequence



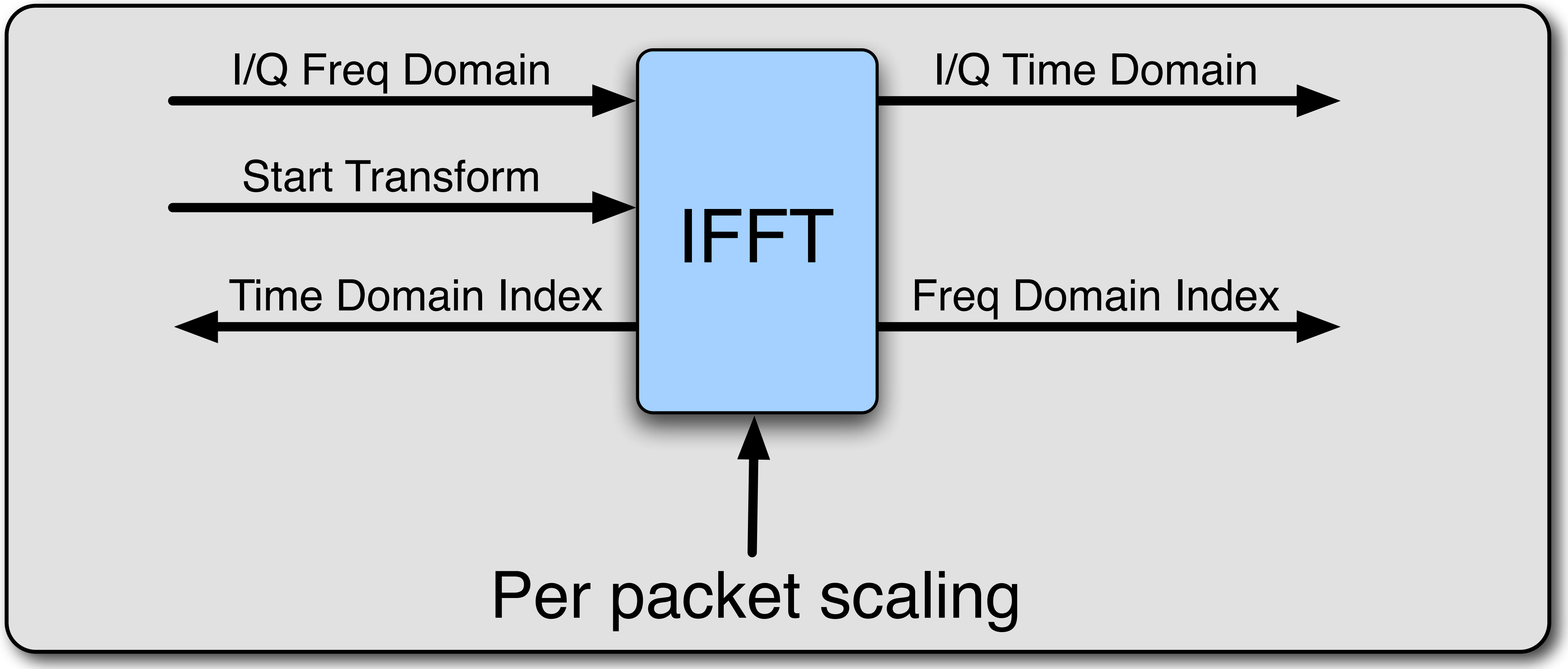
Source Mux Select



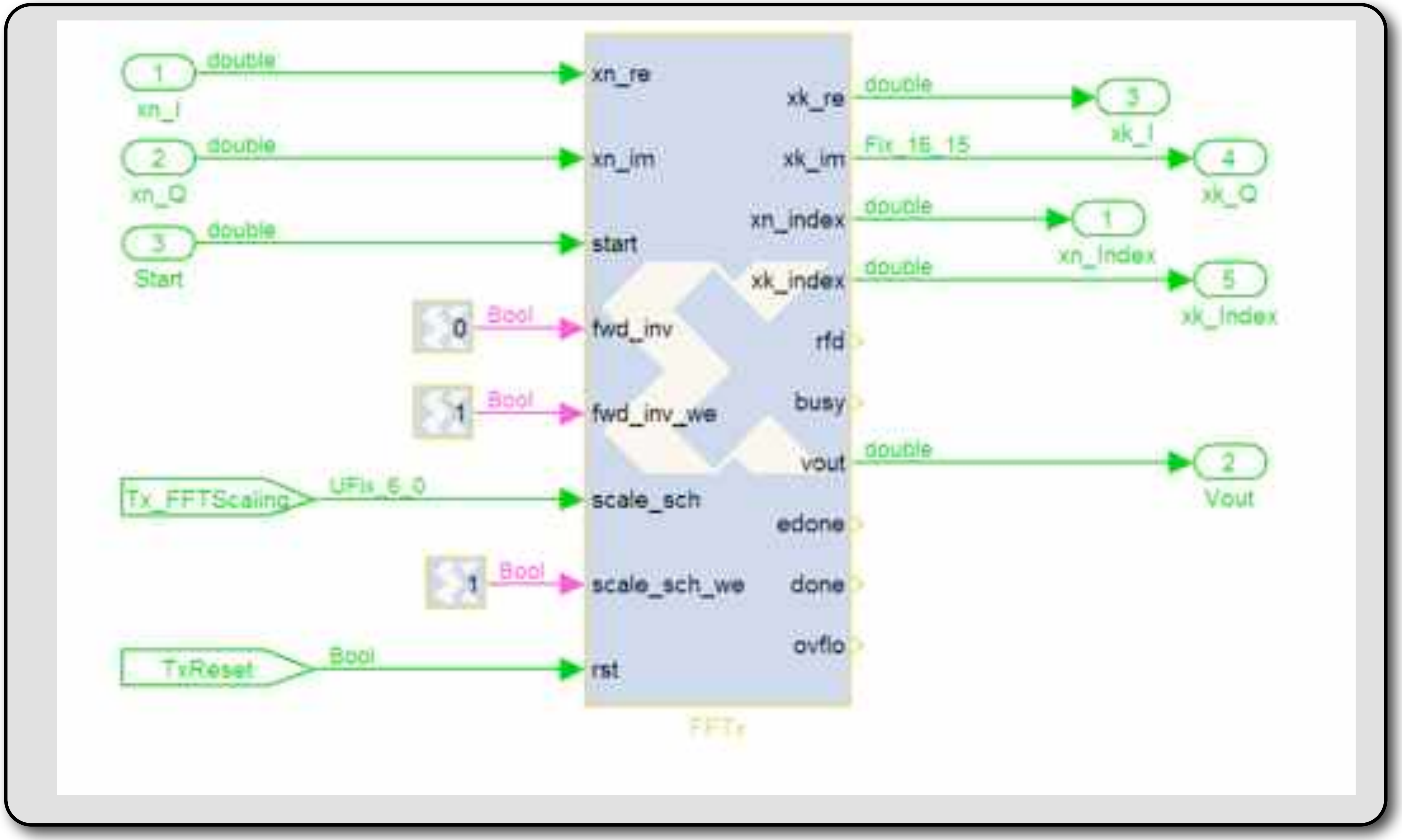
Input IFFT



PHY Example: OFDM Tx



PHY Example: OFDM Tx



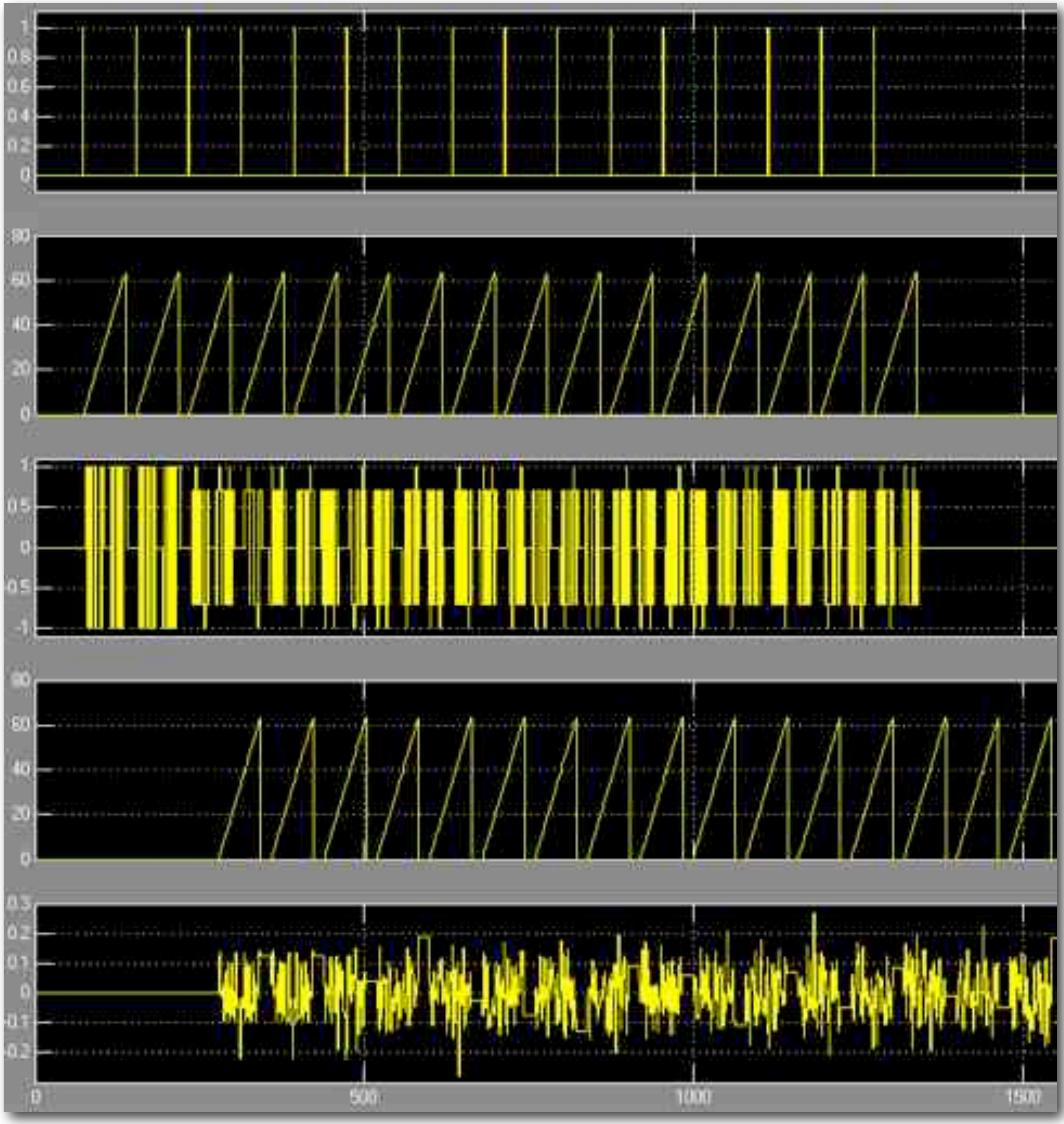
IFFT Start

Input Index

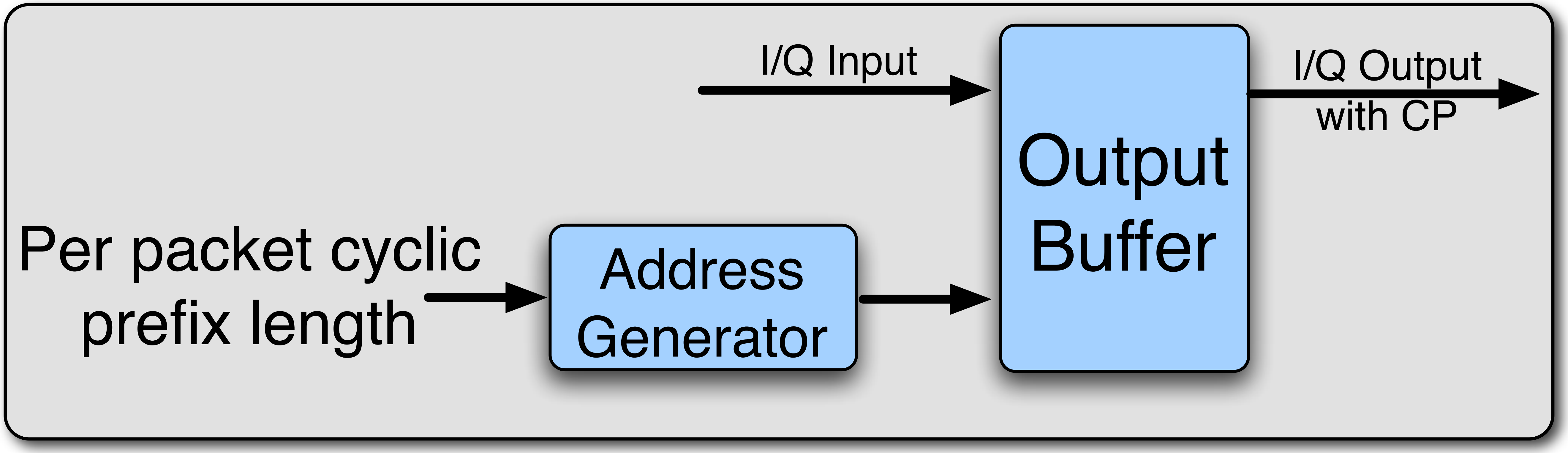
Input Data

Output Index

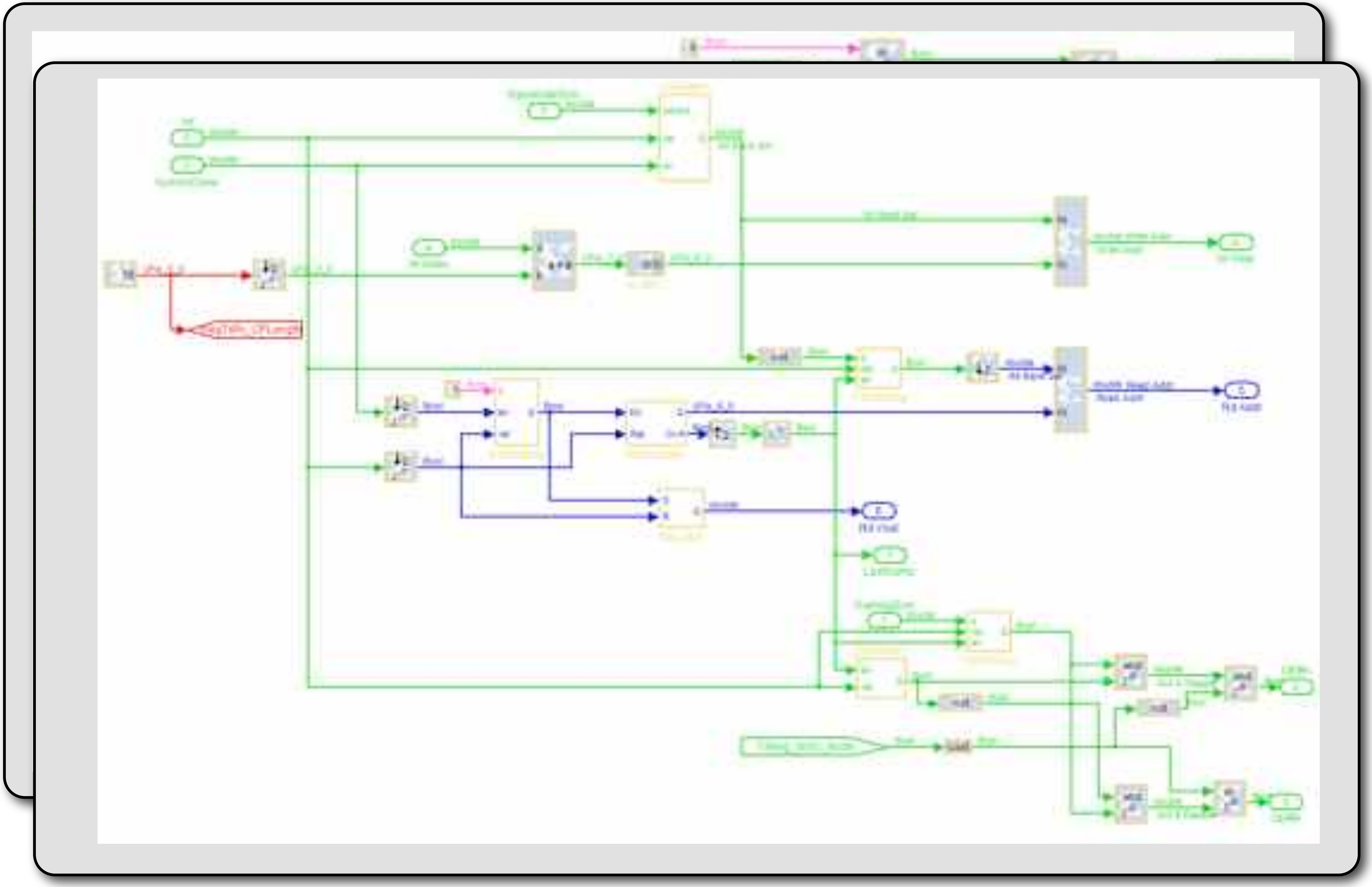
Output Data



PHY Example: OFDM Tx

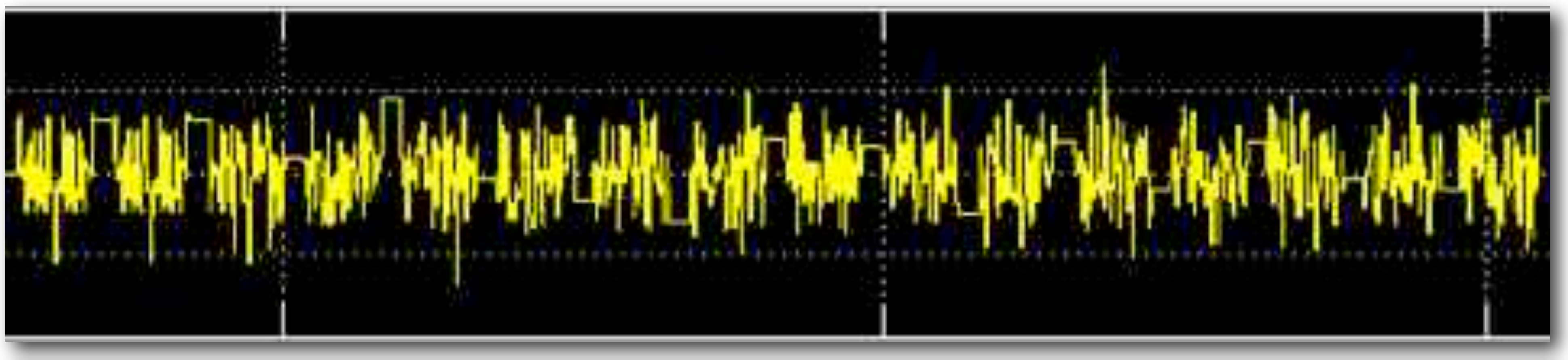


PHY Example: OFDM Tx

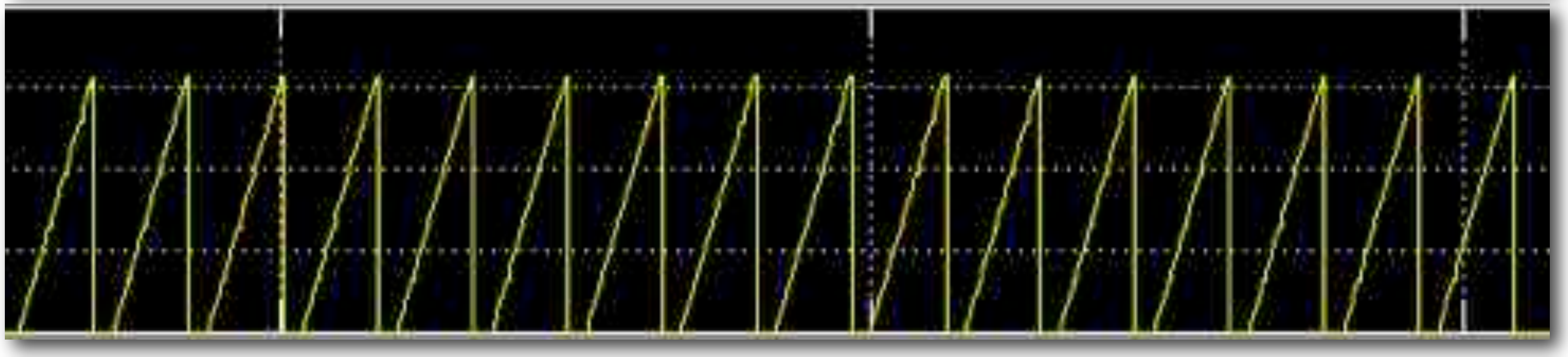


PHY Example: OFDM Tx

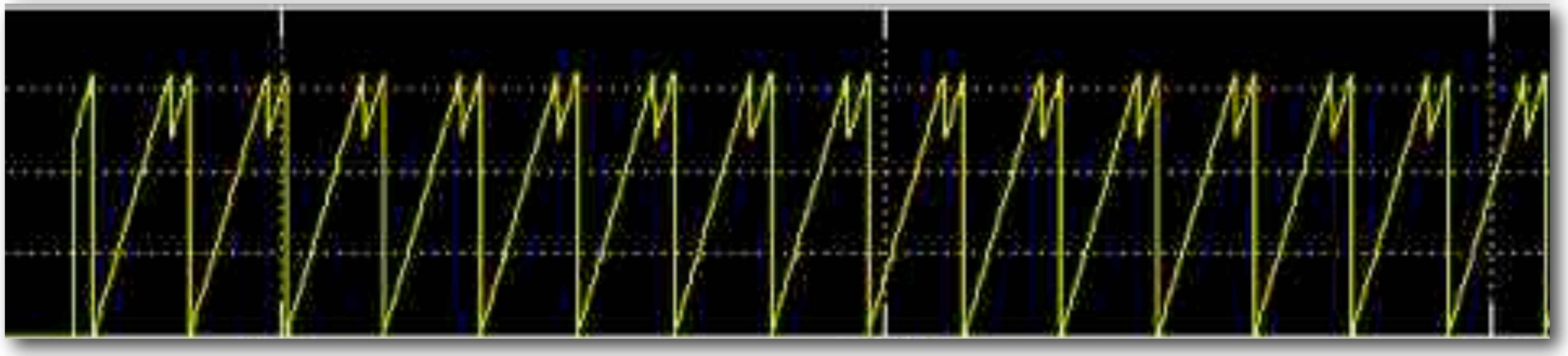
IFFT Output



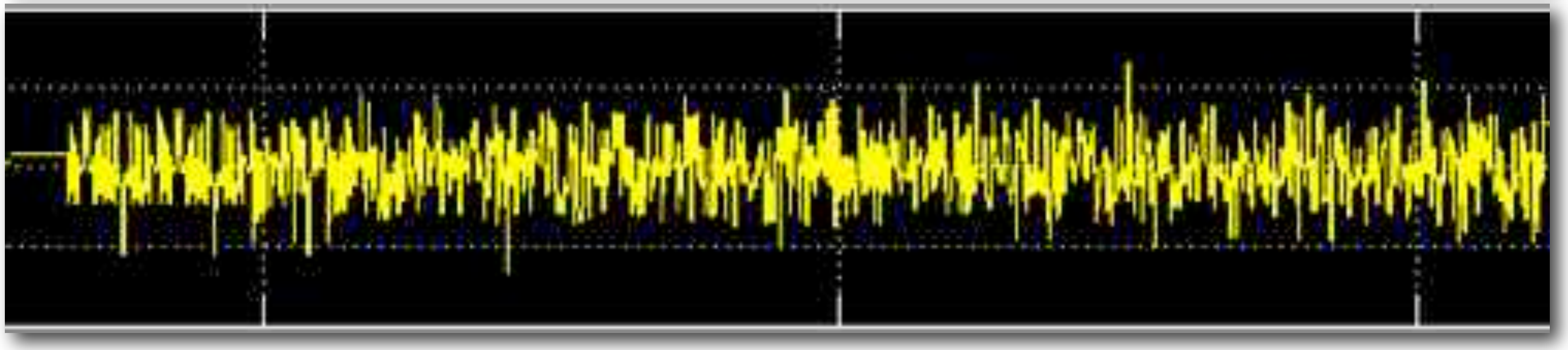
RAM Write Address



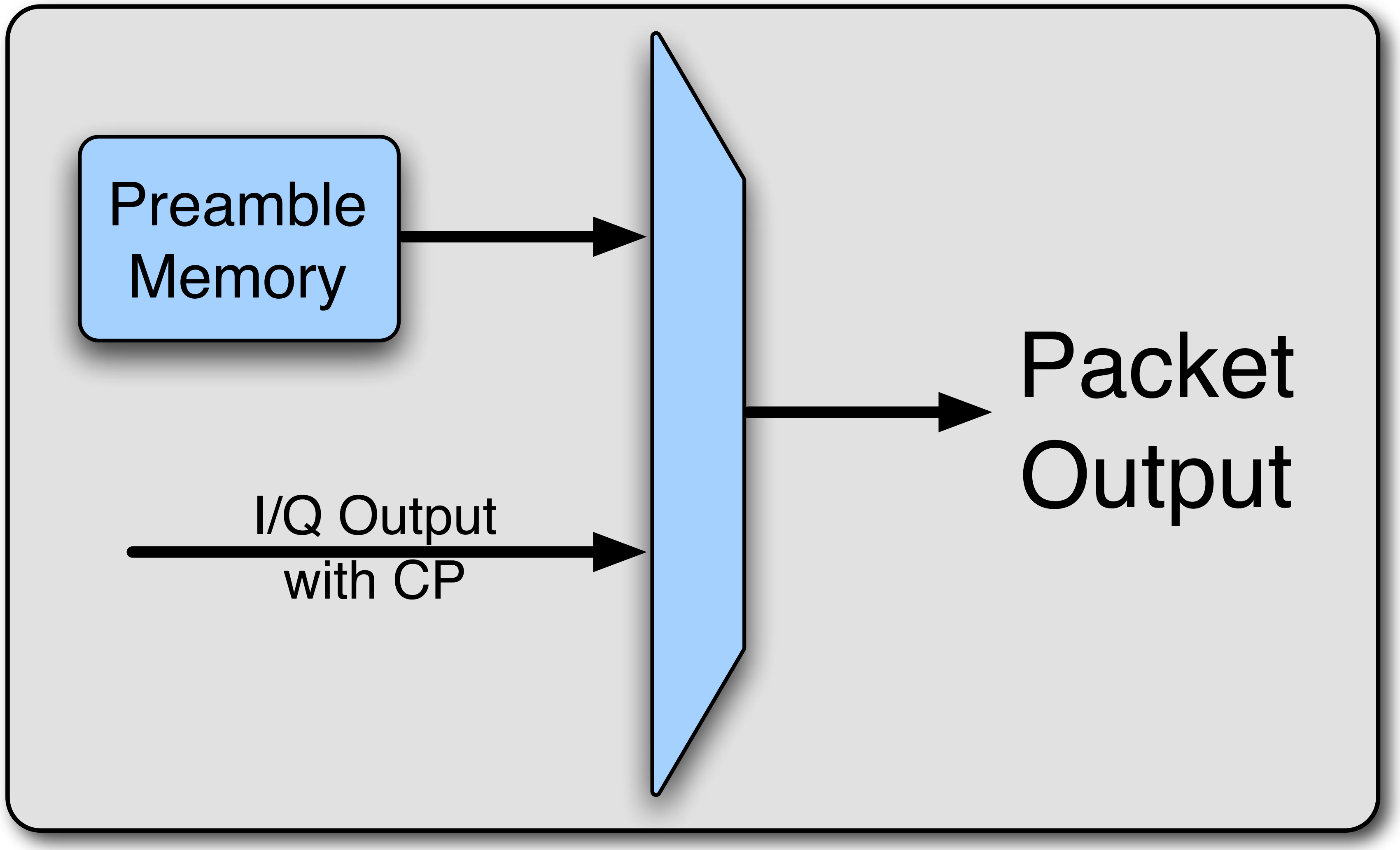
RAM Read Address



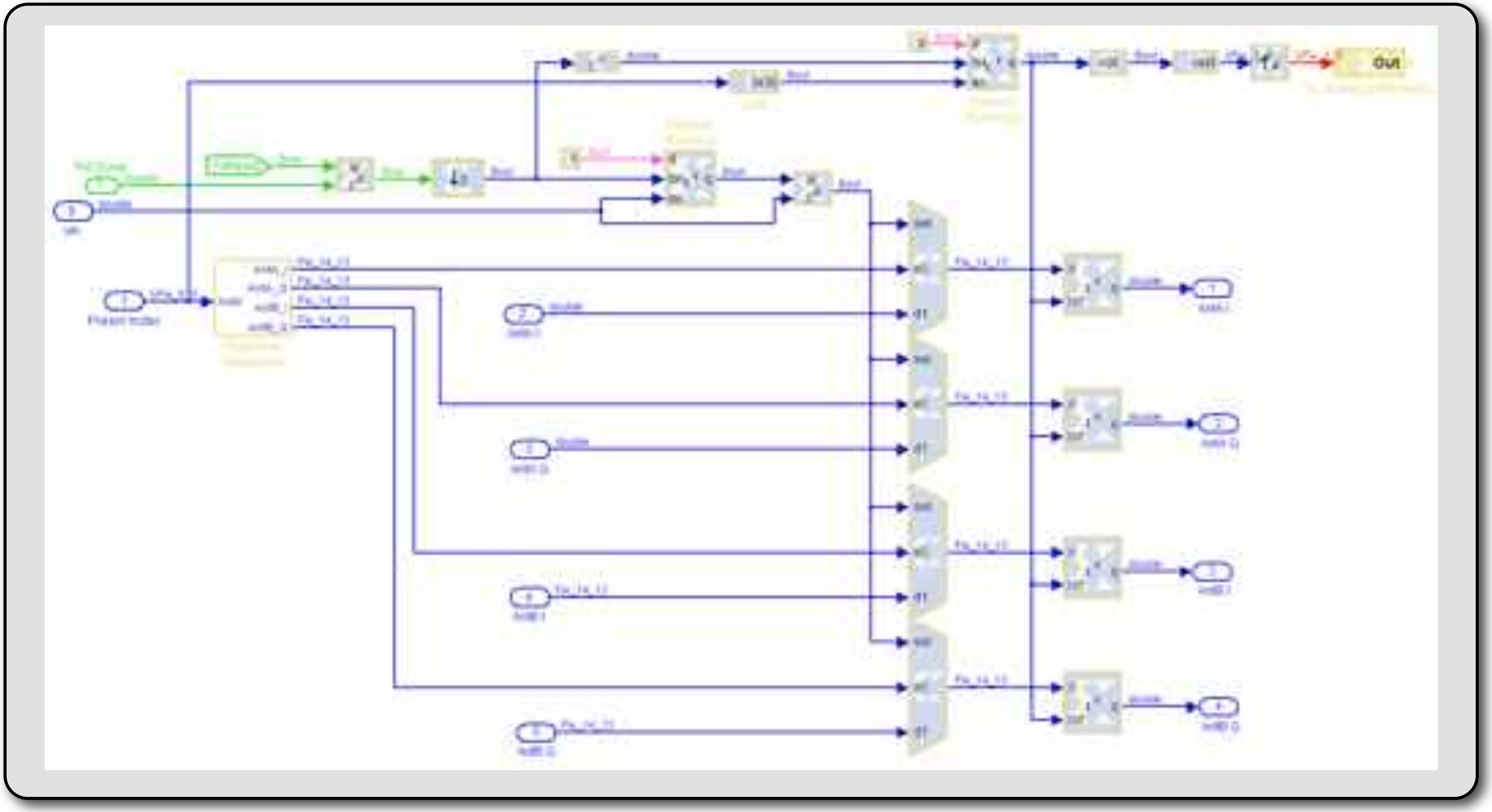
Cyclically Extended



PHY Example: OFDM Tx

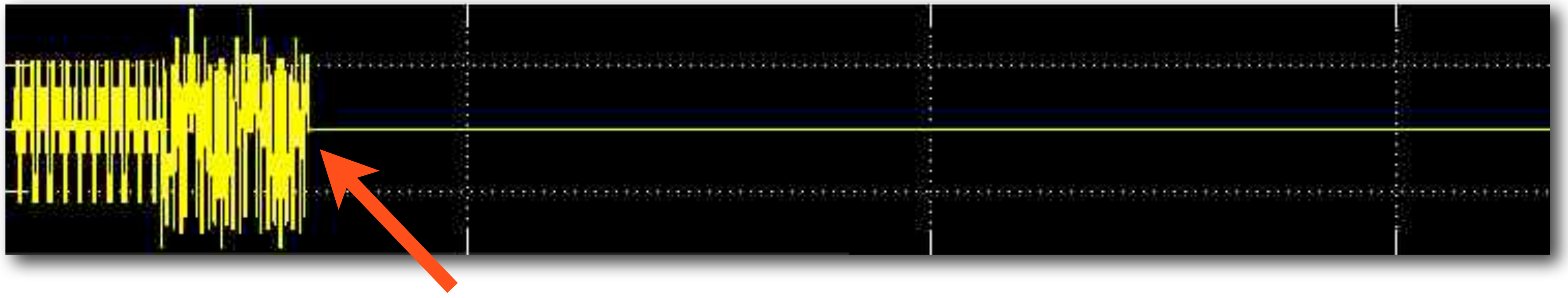


PHY Example: OFDM Tx

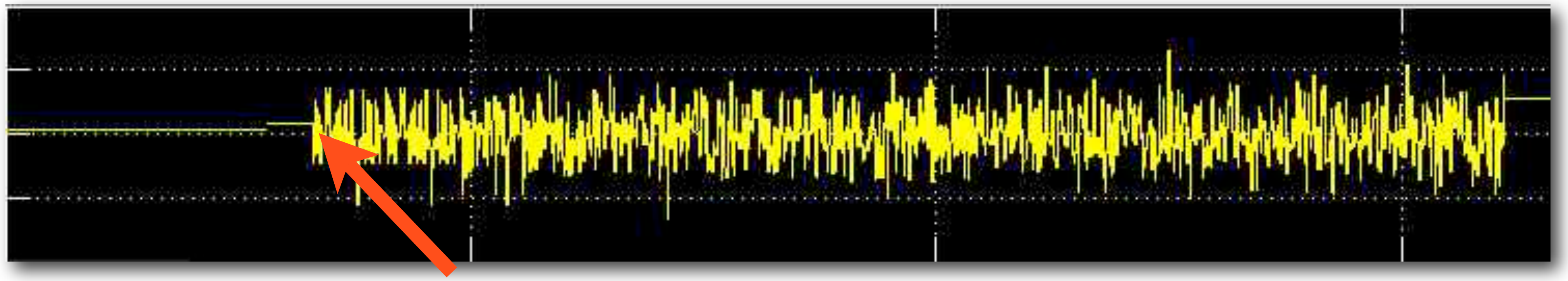


PHY Example: OFDM Tx

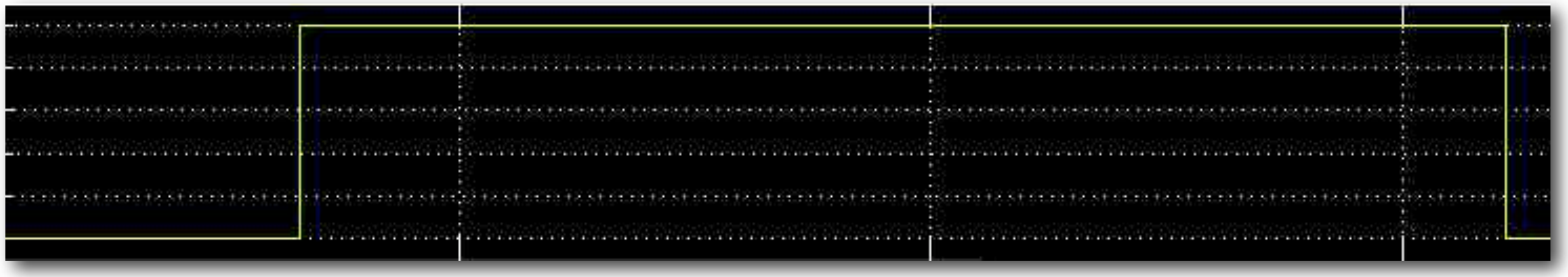
Stored Preamble



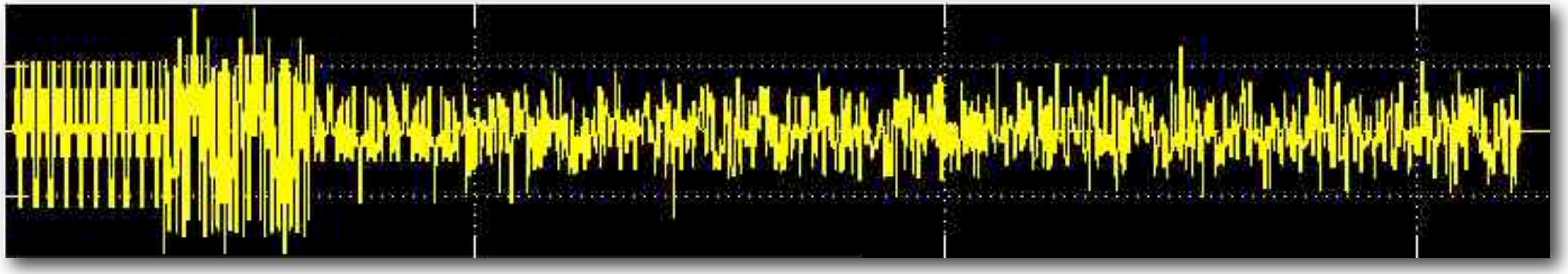
OFDM Output



Output Mux Selection



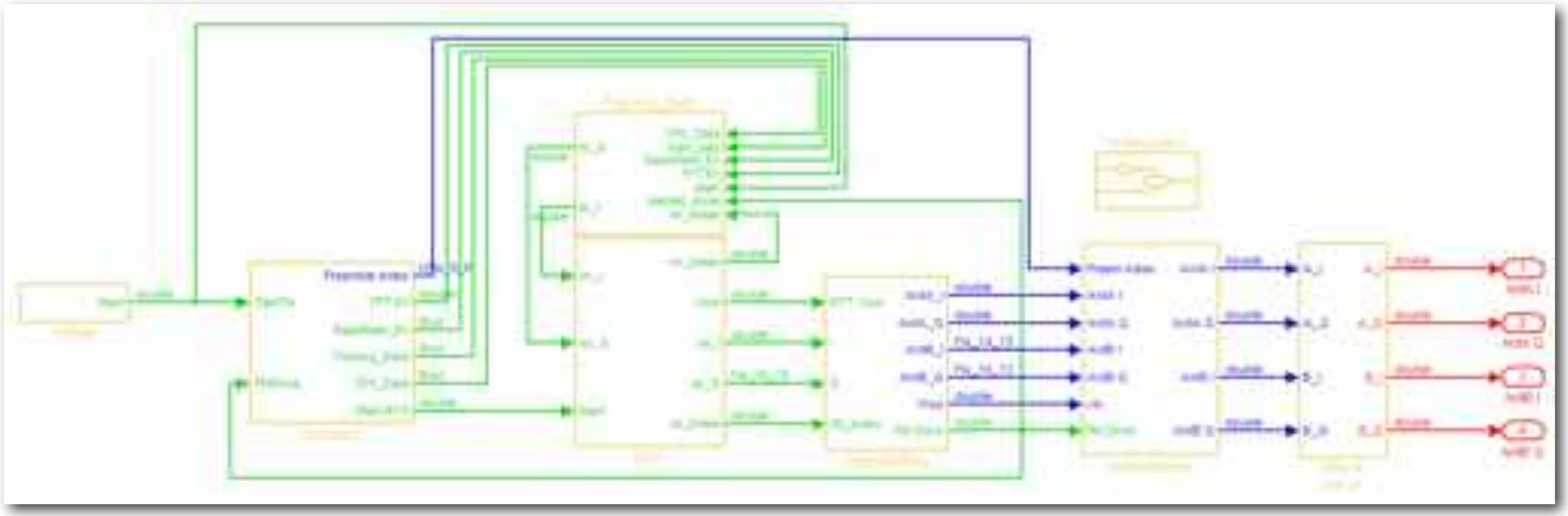
Final Output to DACs



PHY Example: OFDM Tx

- Per packet configuration drives control system
  - Number of training symbols
  - Number of bytes
  - Modulation choices
- Block specific control blocks
  - IFFT start signal
  - Memory address generation
- Triggers & status between core and PowerPC

PHY Example: OFDM Tx

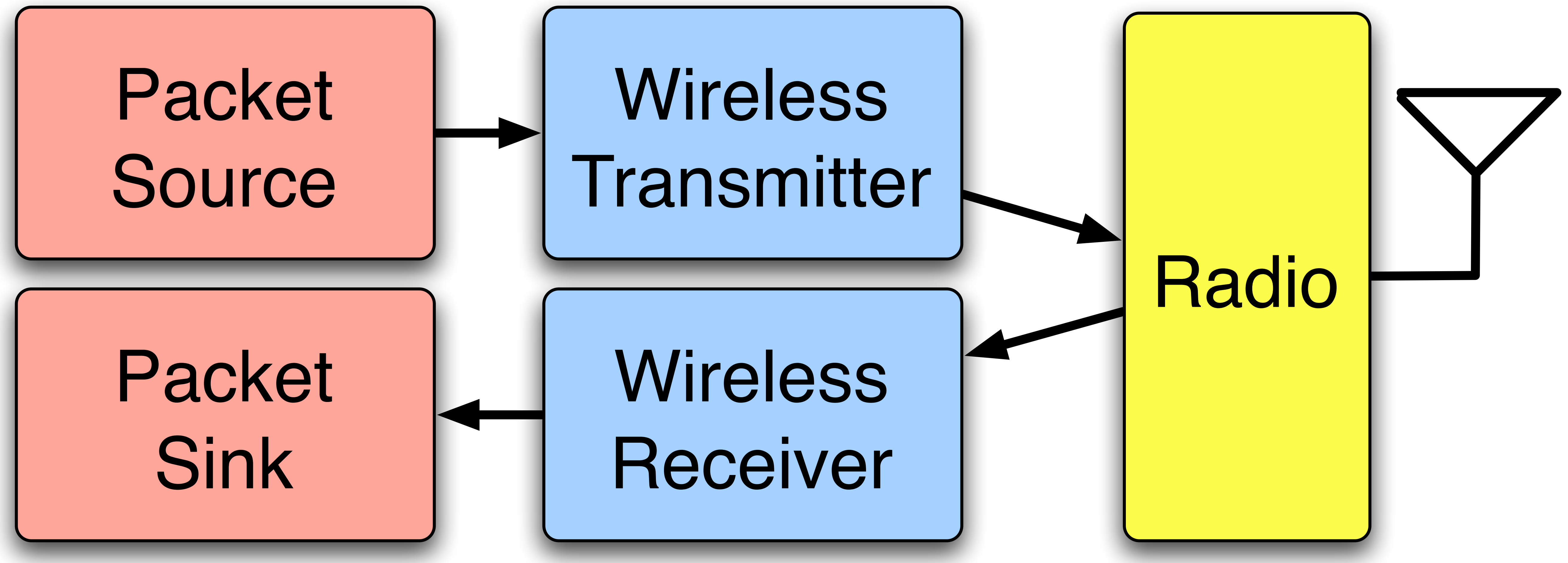


Complete model is available in the WARP repository

Questions?

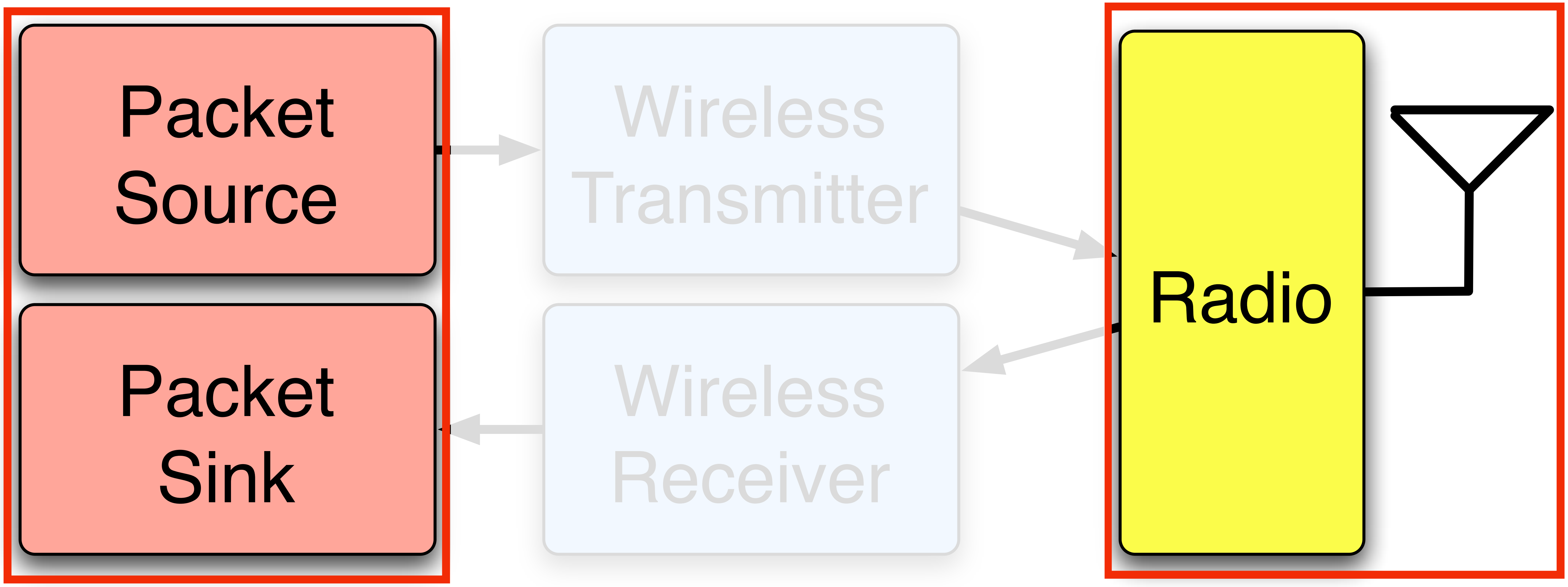
Physical Layer Basics

Simple Wireless Node



Physical Layer Basics

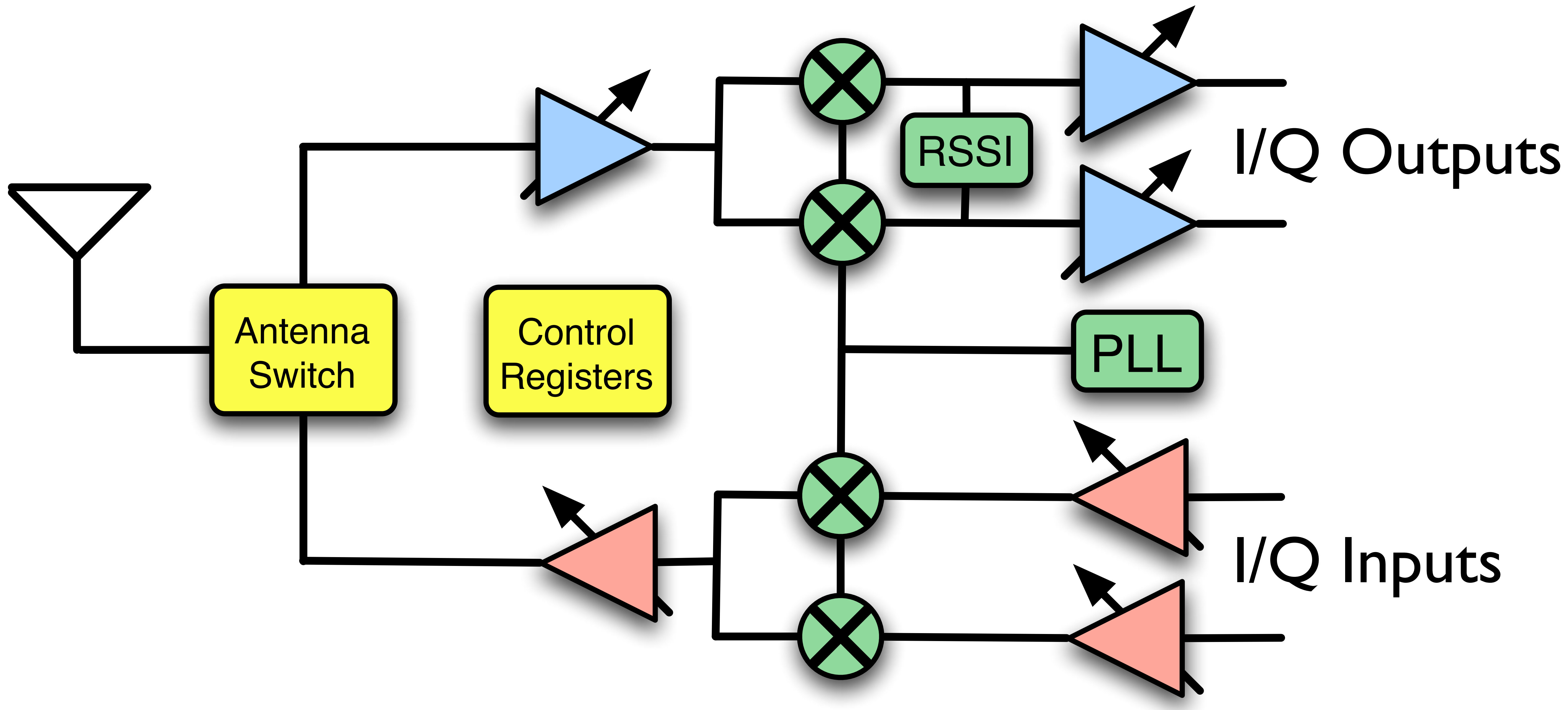
Simple Wireless Node



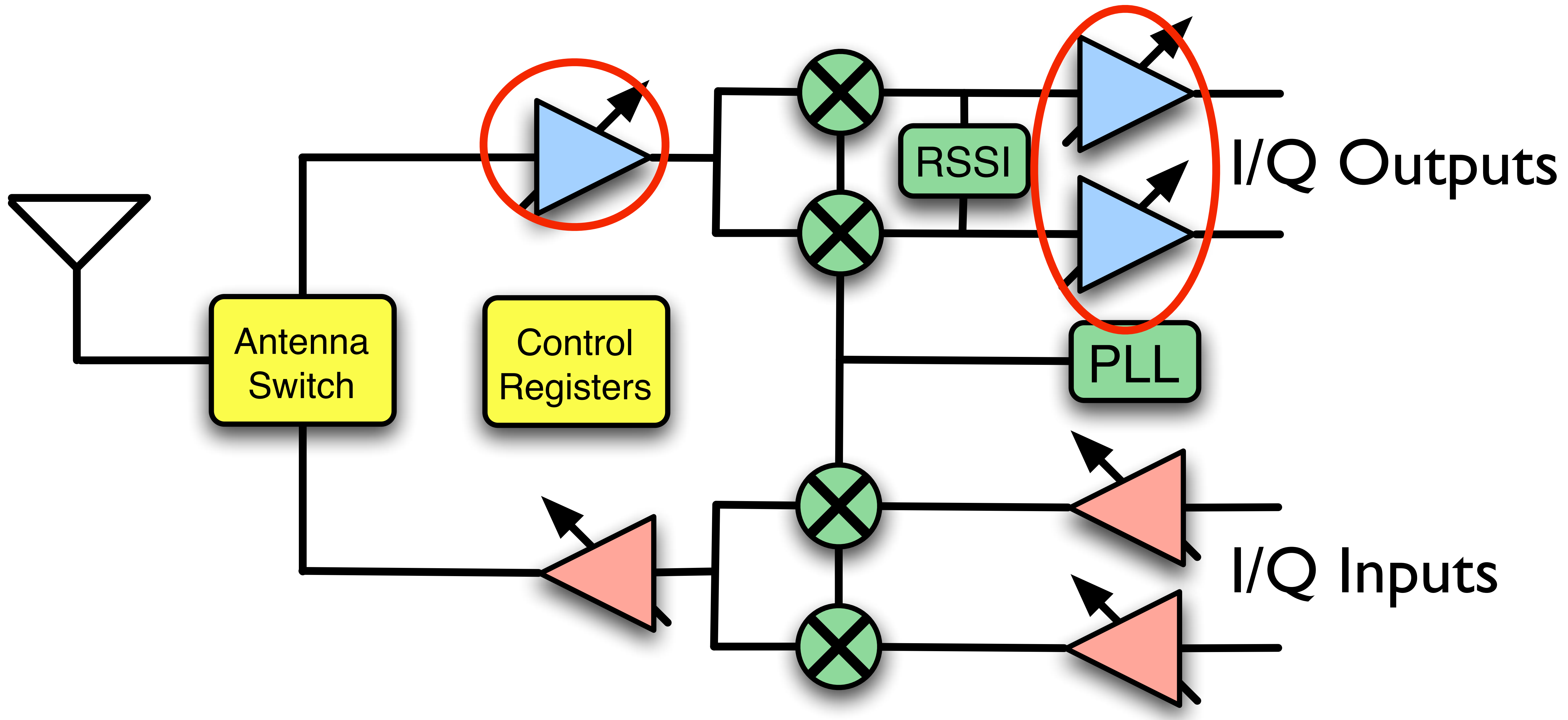
Somebody Else's Problem

Now They're Our Problem

Radio Transceiver

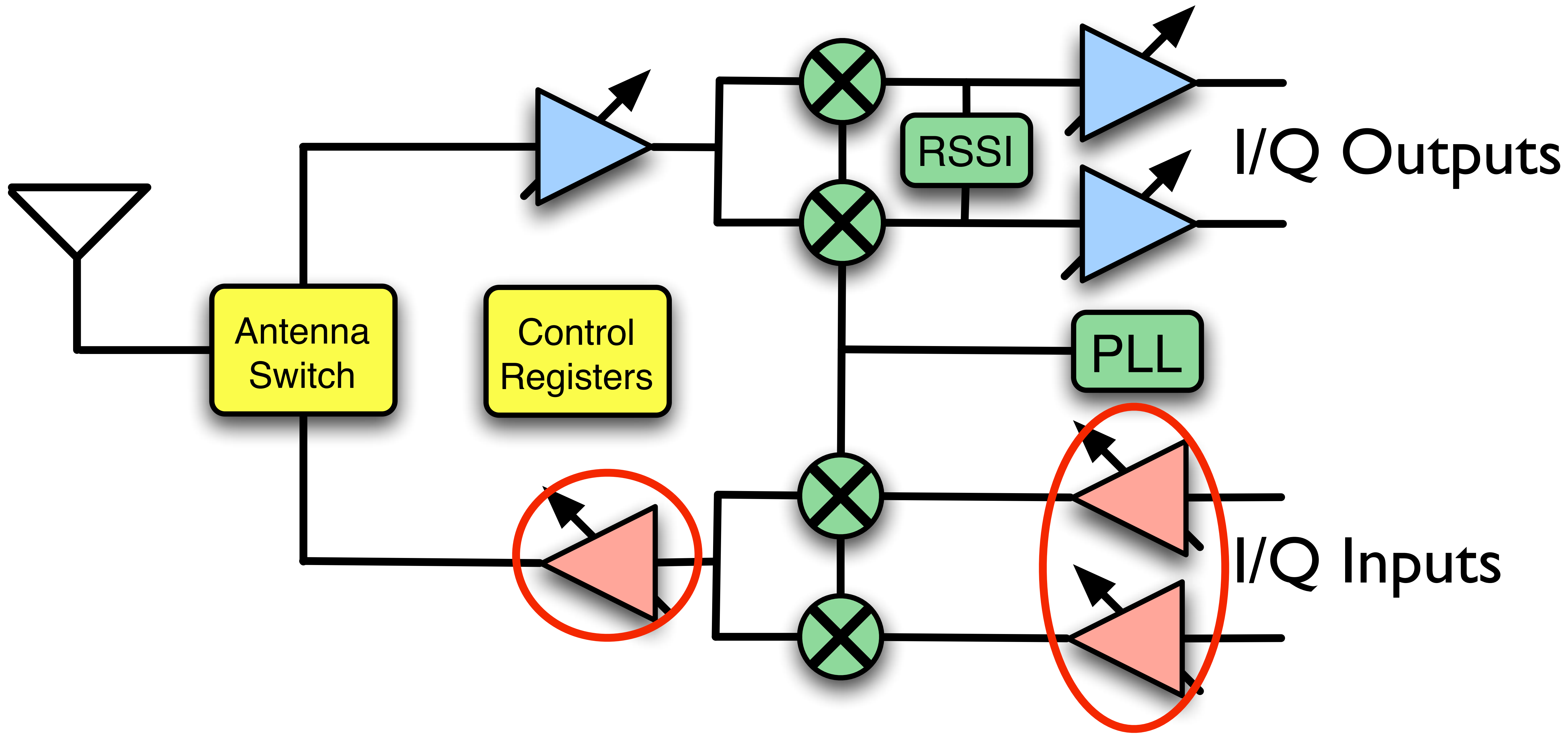


Radio Transceiver



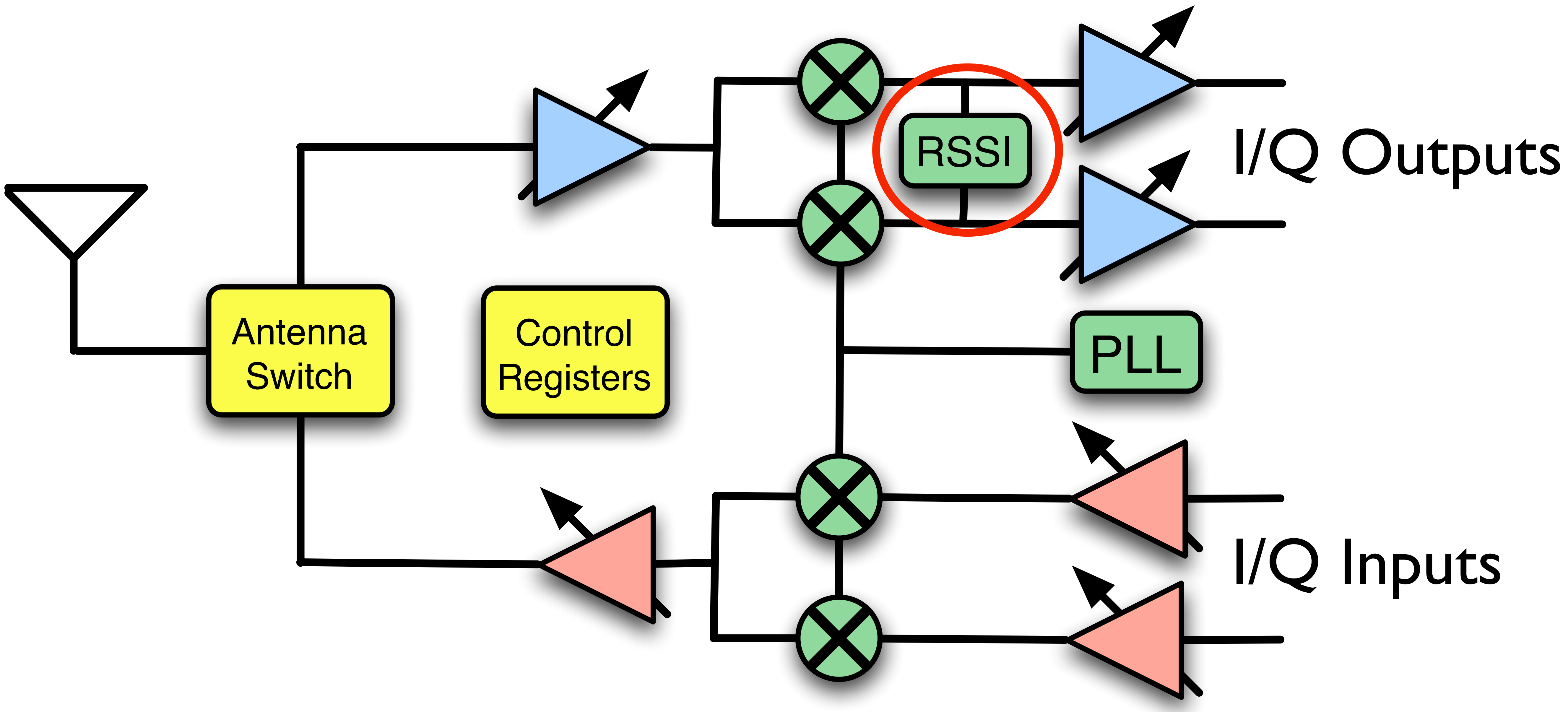
Variable Gain Rx Amplifiers

Radio Transceiver



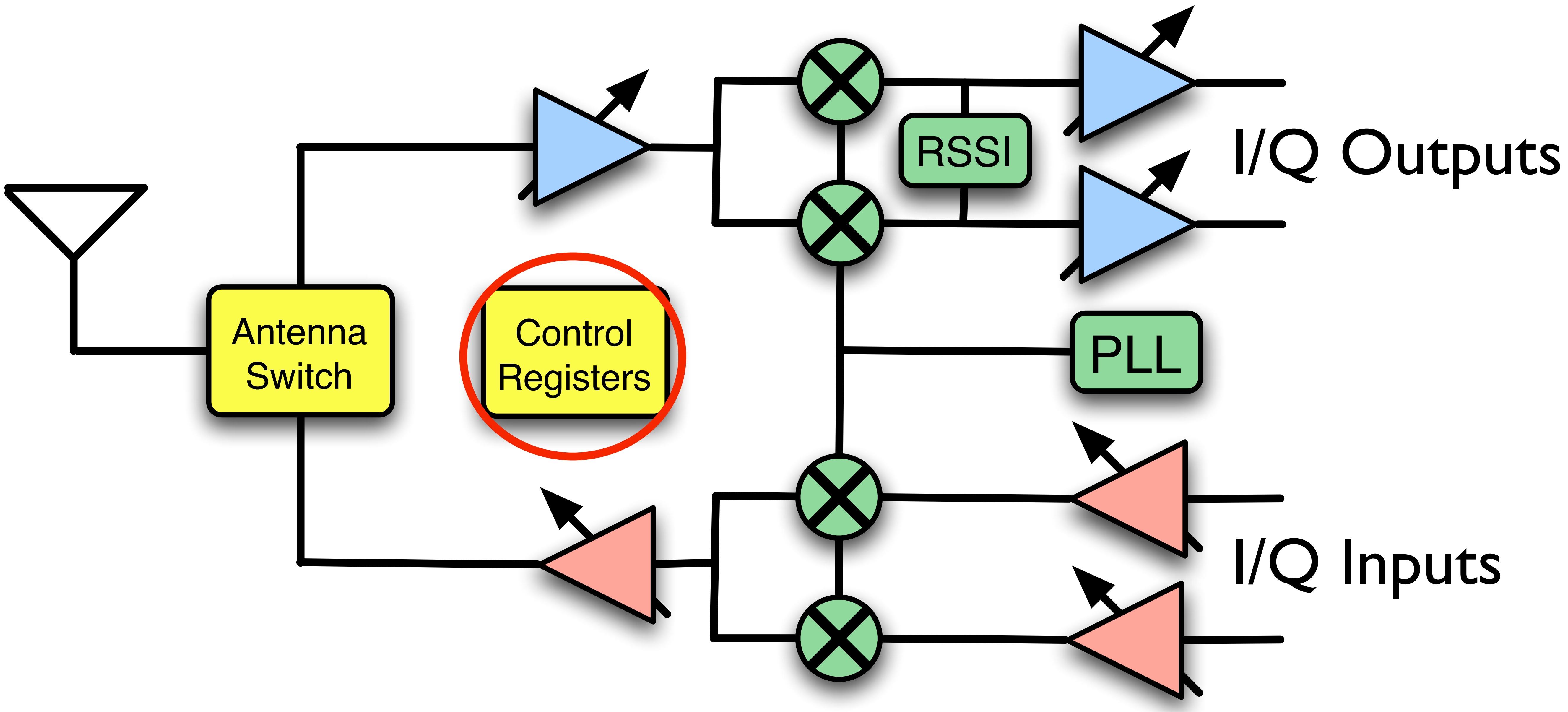
Variable Gain Tx Amplifiers

Radio Transceiver



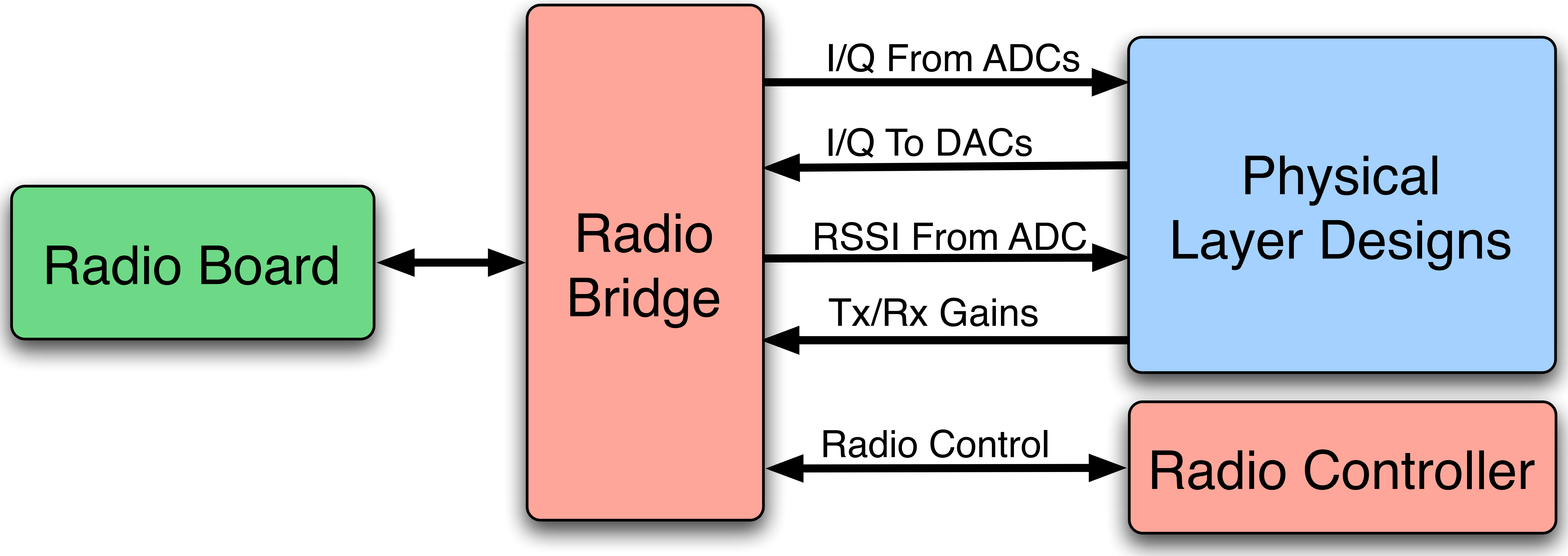
Received Signal Strength Indicator
After RF Gain

Radio Transceiver

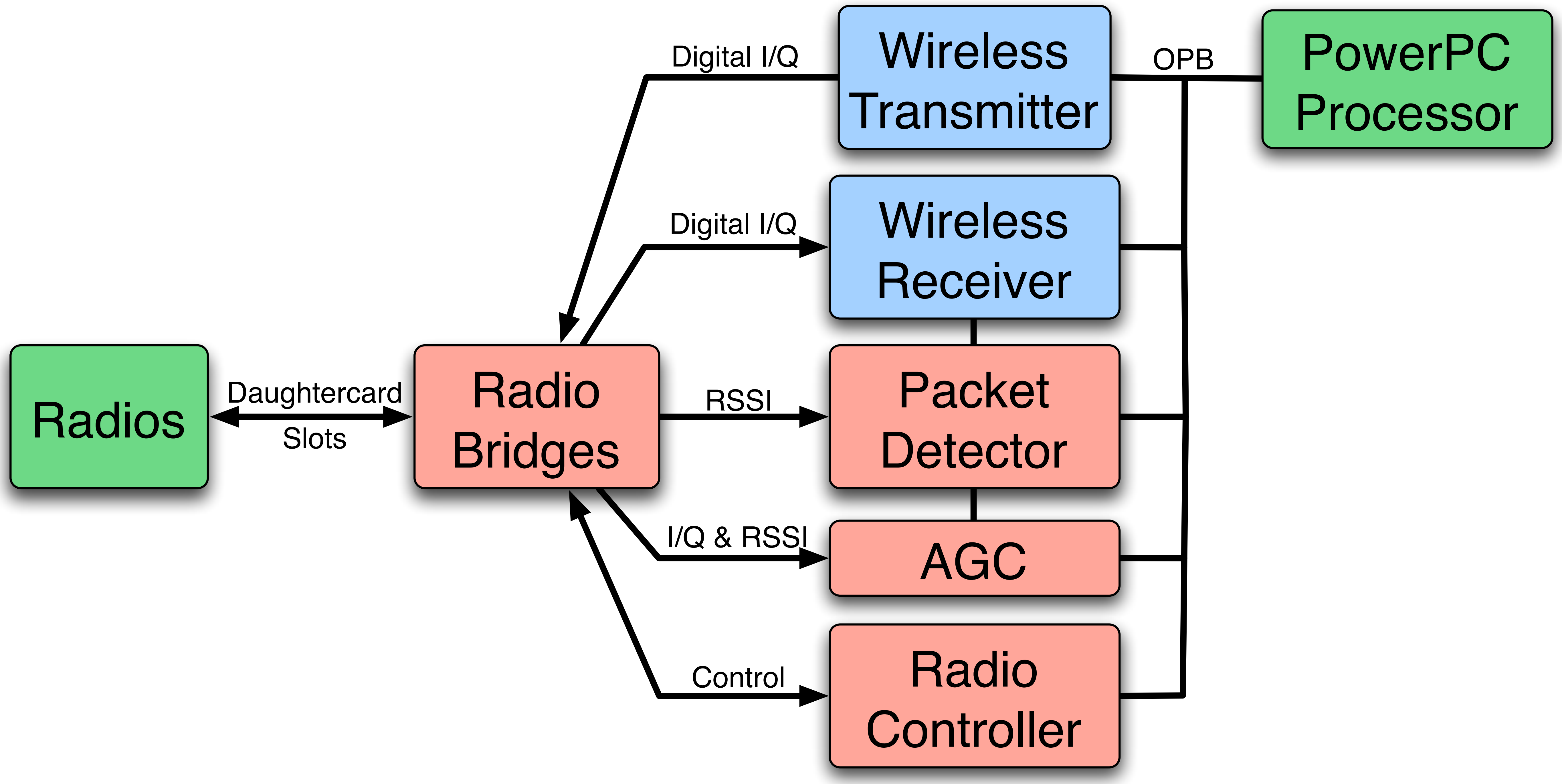


Register Bank
(controlled by SPI interface)

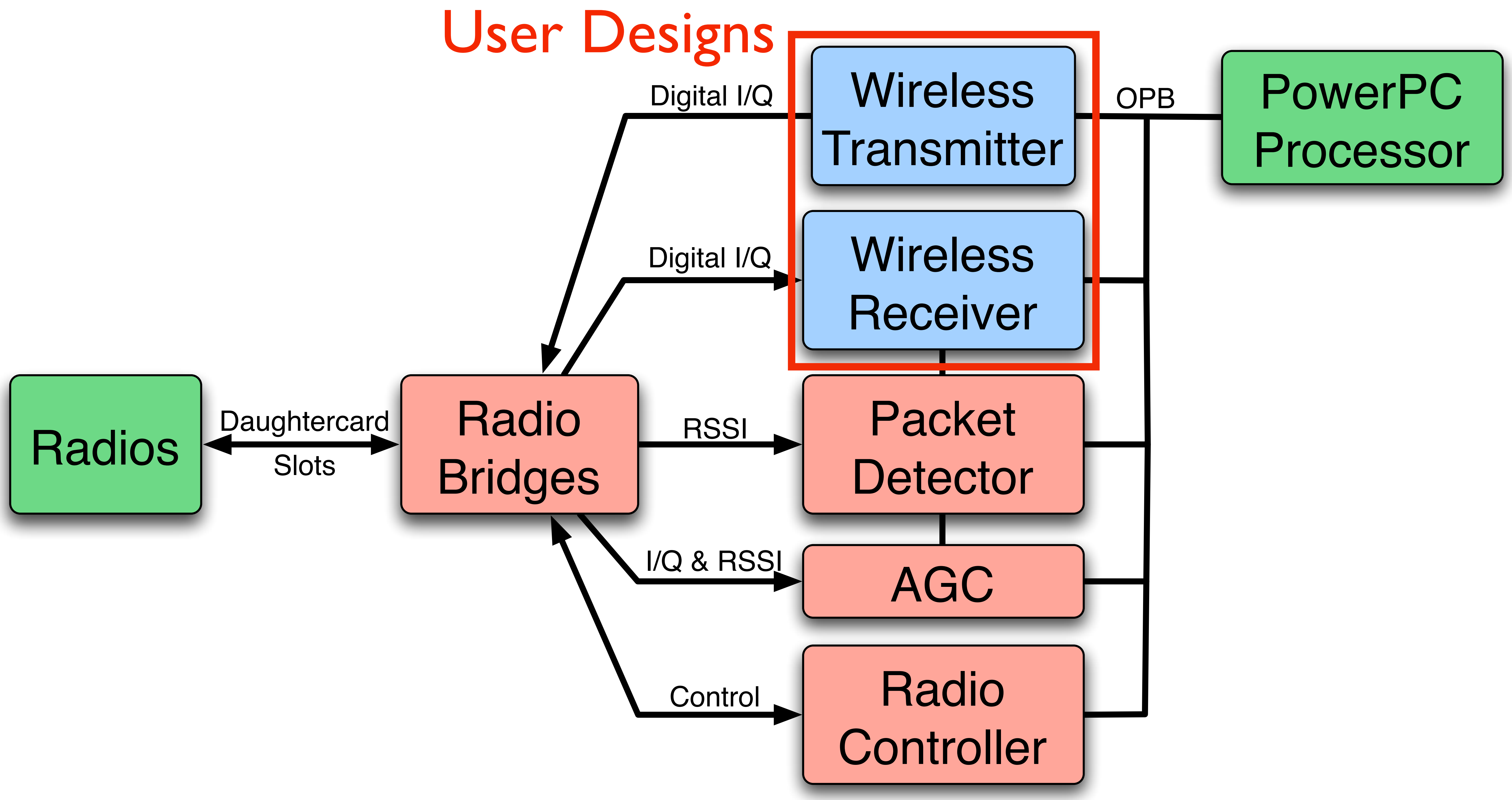
Physical Layer in Hardware



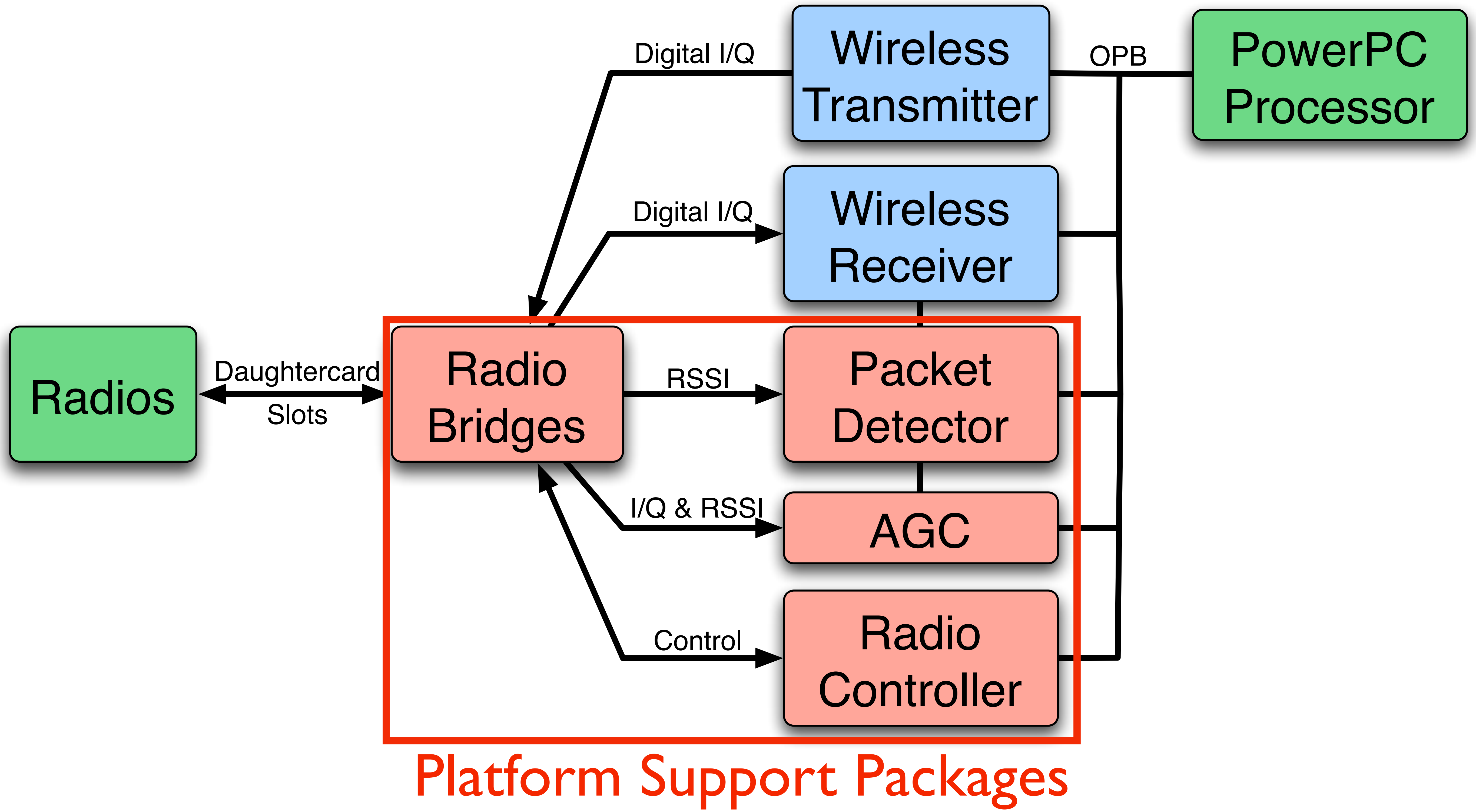
Physical Layer in Hardware



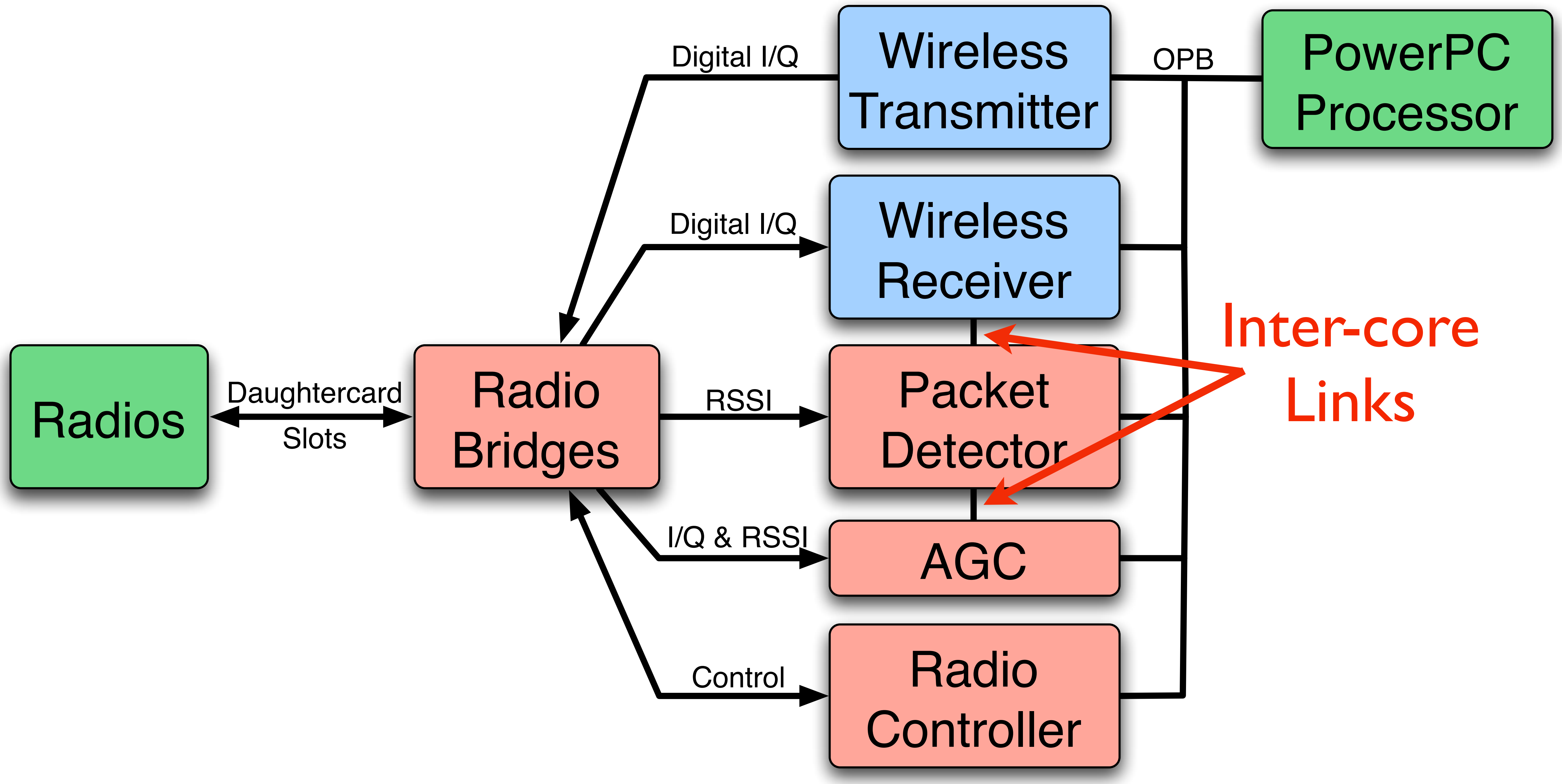
Physical Layer in Hardware



Physical Layer in Hardware

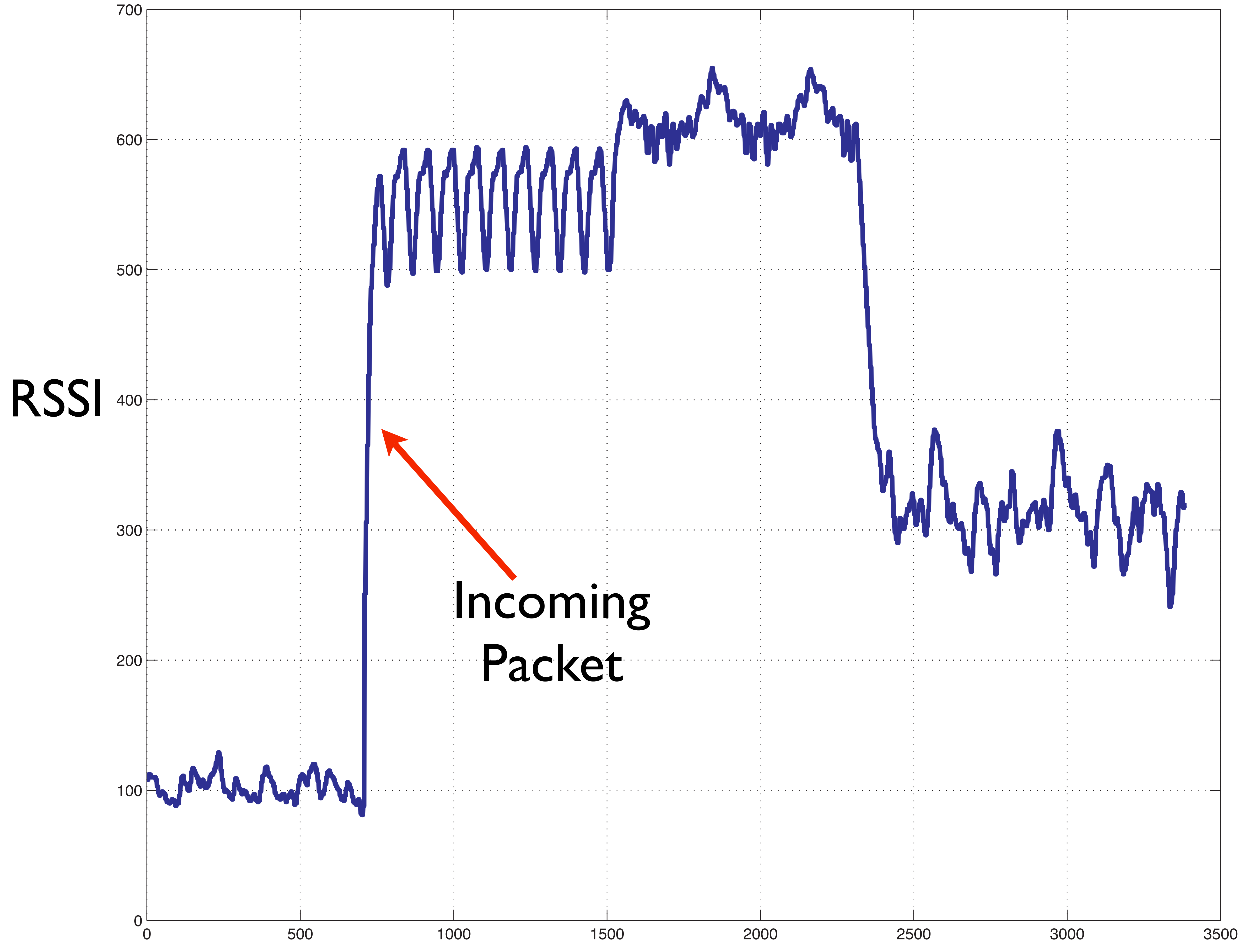


Physical Layer in Hardware



Packet Detection

- Triggers AGC & receiver models
- Detection based only on received energy
  - I/Q saturated and too corrupted
  - Gain adjusted *after* detection
- Detection confirmed/rejected by Rx PHY
  - Requires some data-aided detection
  - Correlates against every packet's preamble


RSSI
Incoming
Packet
700
600
500
400
300
200
100
0
0
500
1000
1500
2000
2500
3000
3500

Automatic Gain Control

- Receiver has 90 dB gain range
  - RF gain of 0, 15 or 30 dB
  - Baseband gain of 0...60 dB
- Amplifiers start max gain with each packet
- AGC reduces gain in first 5 µs
  - RF gain set by RSSI
  - Baseband gain set by I/Q averages

Radio Controller

- Controller hardware
  - I/O registers & SPI controller
  - One core controls all 4 radios & DACs
- Controller software
  - Full C API for radio board control
  - All radio features controlled by C functions
  - Simple functions required
  - Advanced functions optional

Radio Controller API

```
WarpRadio_v1_Reset()

WarpRadio_v1_TxEnable()
WarpRadio_v1_SetCenterFreq2GHz()
WarpRadio_v1_BaseBandTxGain()
WarpRadio_v1_TxVGAGainControl()
WarpRadio_v1_24AmpEnable()

WarpRadio_RxEnable()
WarpRadio_RxLNAGainControl()
WarpRadio_RxVGAGainControl()
WarpRadio_RxLpfCornFreqCoarseAdj()
```

Radio Controller API

```
WarpRadio_v1_Reset()

WarpRadio_v1_ TxEnable()
WarpRadio_v1_ SetCenterFreq2GHz()
WarpRadio_v1_ BaseBandTxGain()
WarpRadio_v1_ TxVGAGainControl()
WarpRadio_v1_ 24AmpEnable()


WarpRadio_RxEnable()
WarpRadio_RxLNAGainControl()
WarpRadio_RxVGAGainControl()
WarpRadio_RxLpfCornFreqCoarseAdj()
```

Full API online:

http://warp.rice.edu/WARP_API

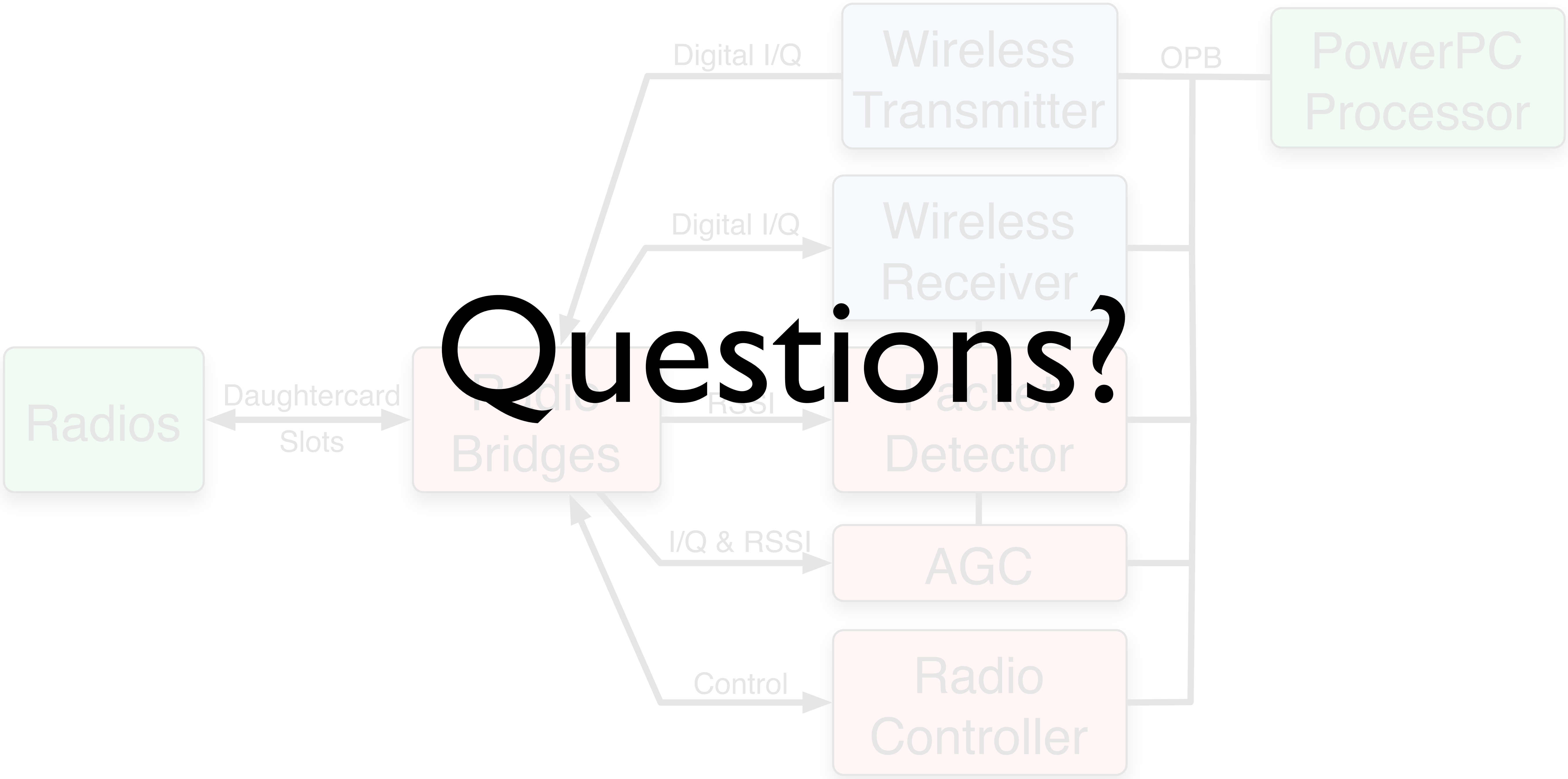
Radio Bridge

- Ties user designs to radio hardware
  - Ports for user signals (ADC, DAC, gains)
  - Ports for radio controller I/O
- Users instantiate one bridge per radio board
- All constraints & most links are automatic
- Custom Verilog peripheral

PHY Design Review

- Build & verify PHY in FPGA design tool
  - System Generator is a good choice
  - Make sure everything works in simulation
- Generate simple Tx/Rx peripherals
  - "Cheating" is good at first
- Hook up your core in the EDK
  - Use our radio bridges & controller
- Generate the platform & test it in hardware

Questions?


Radios
Daughtercard Slots
Radio Bridges
Digital I/Q
Wireless Transmitter
Digital I/Q
Wireless Receiver
RSSI
Packet Detector
I/Q & RSSI
AGC
Control
Radio Controller
OPB
PowerPC Processor

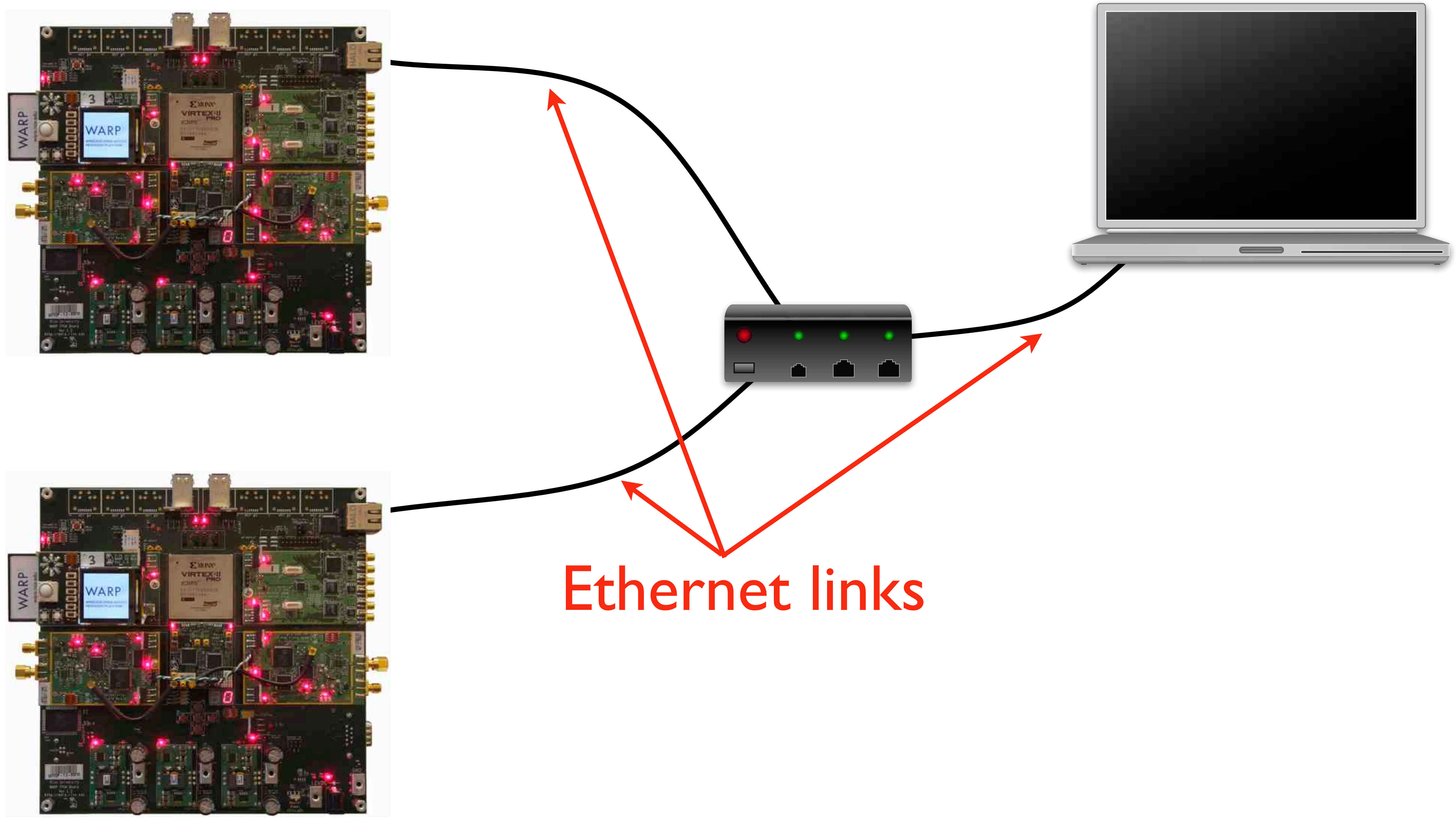
PHY Design Flows

- Real-time PHY design
  - Low-level FPGA design
  - Putting it all together
- WARPLab
  - MATLAB↔WARP Link
  - Very rapid prototyping of PHY algorithms

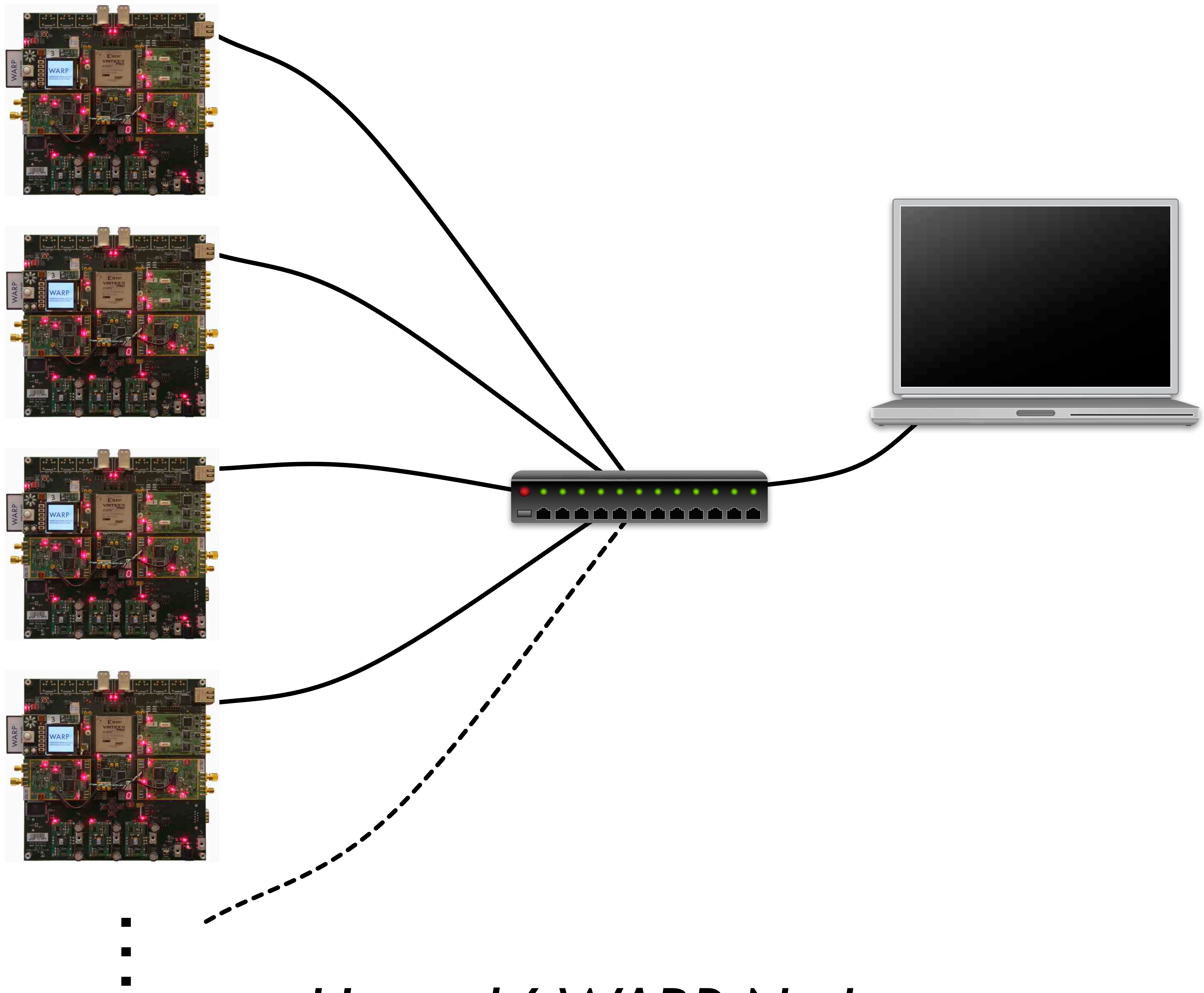
WARPLab Overview

- MATLAB for signal processing
- WARP for wireless interfaces
- Real-time channel use
- Non-real-time processing
- One PC controls many WARP nodes

WARPLab Overview

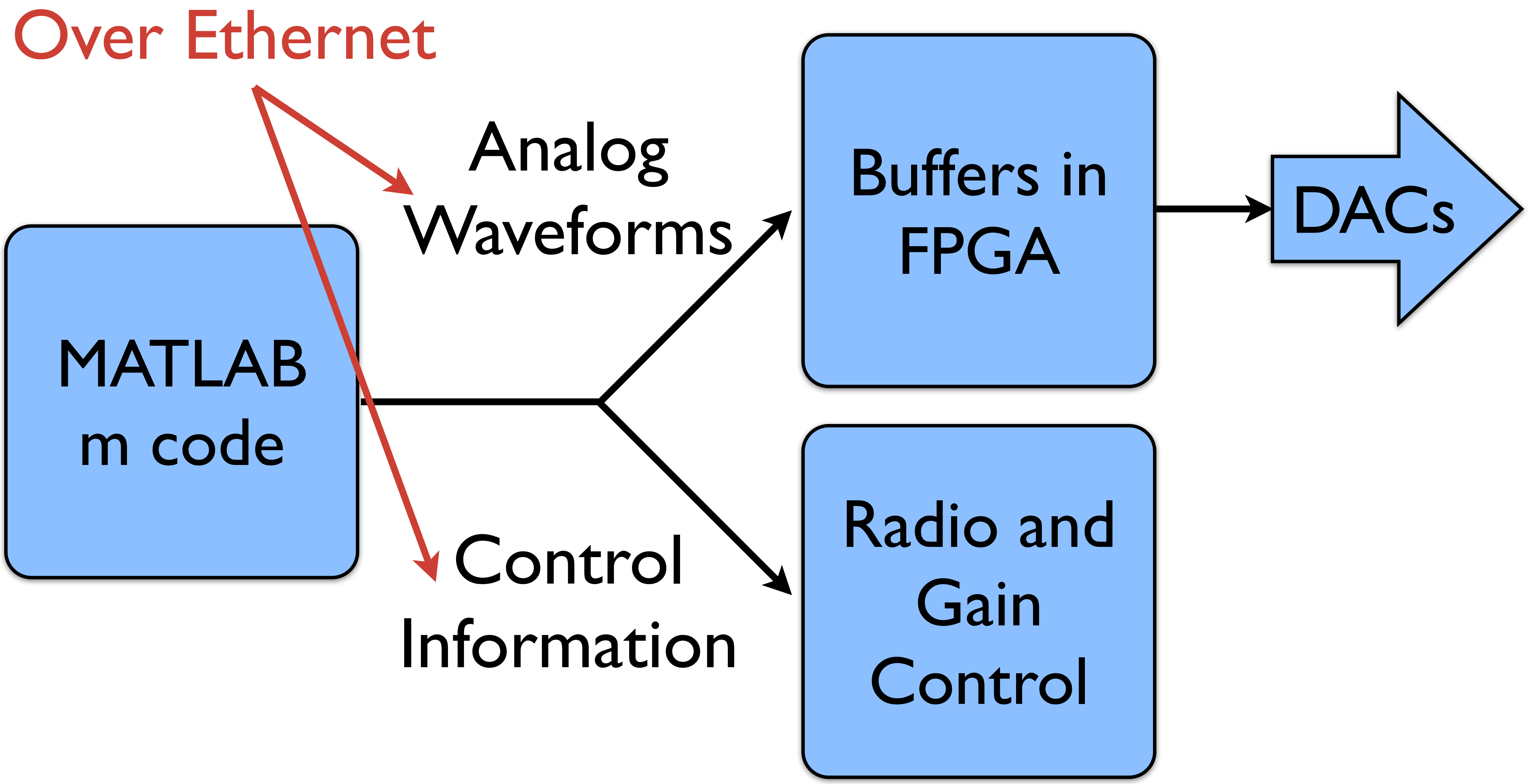


WARPLab Overview

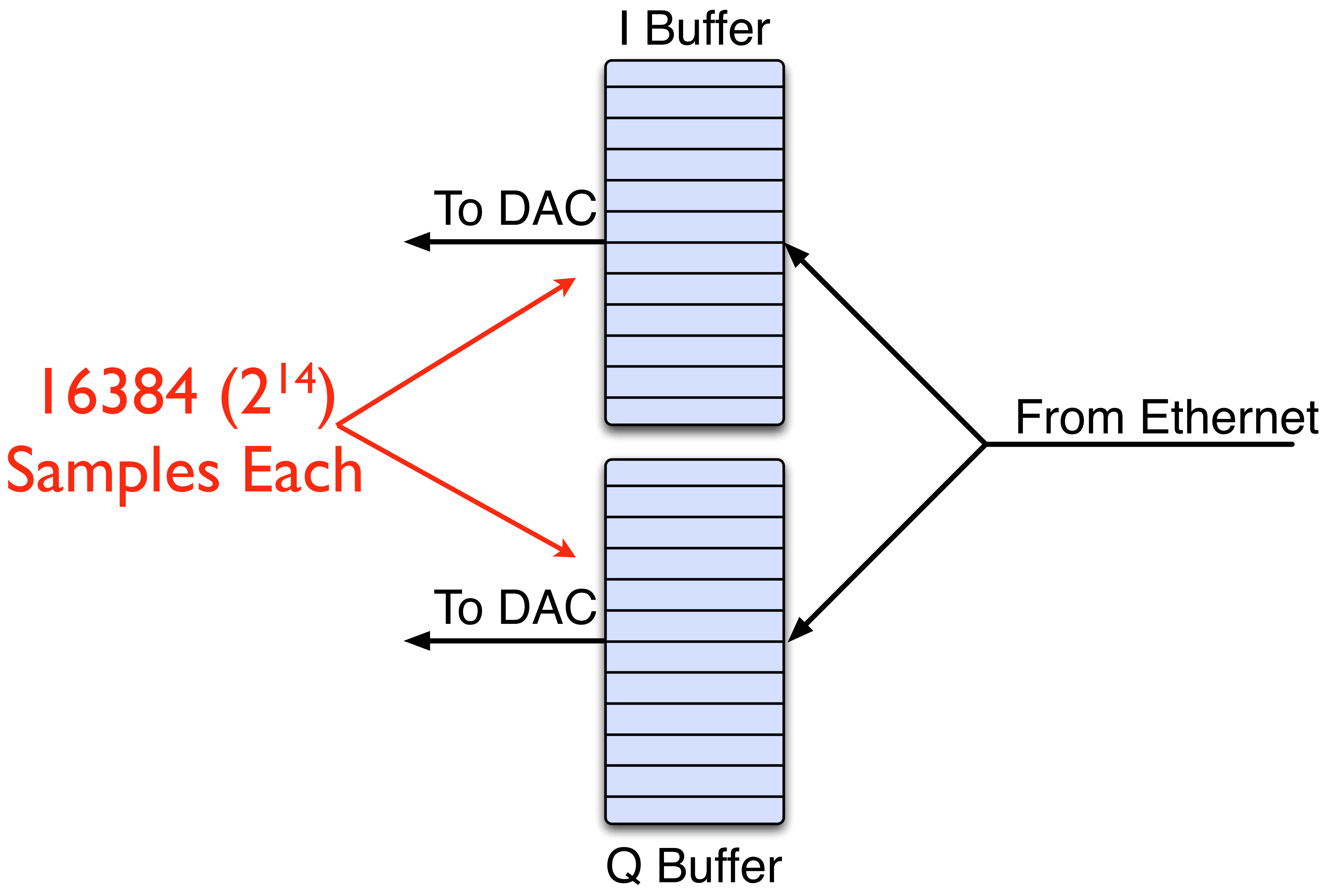


Up to 16 WARP Nodes

WARPLab Architecture

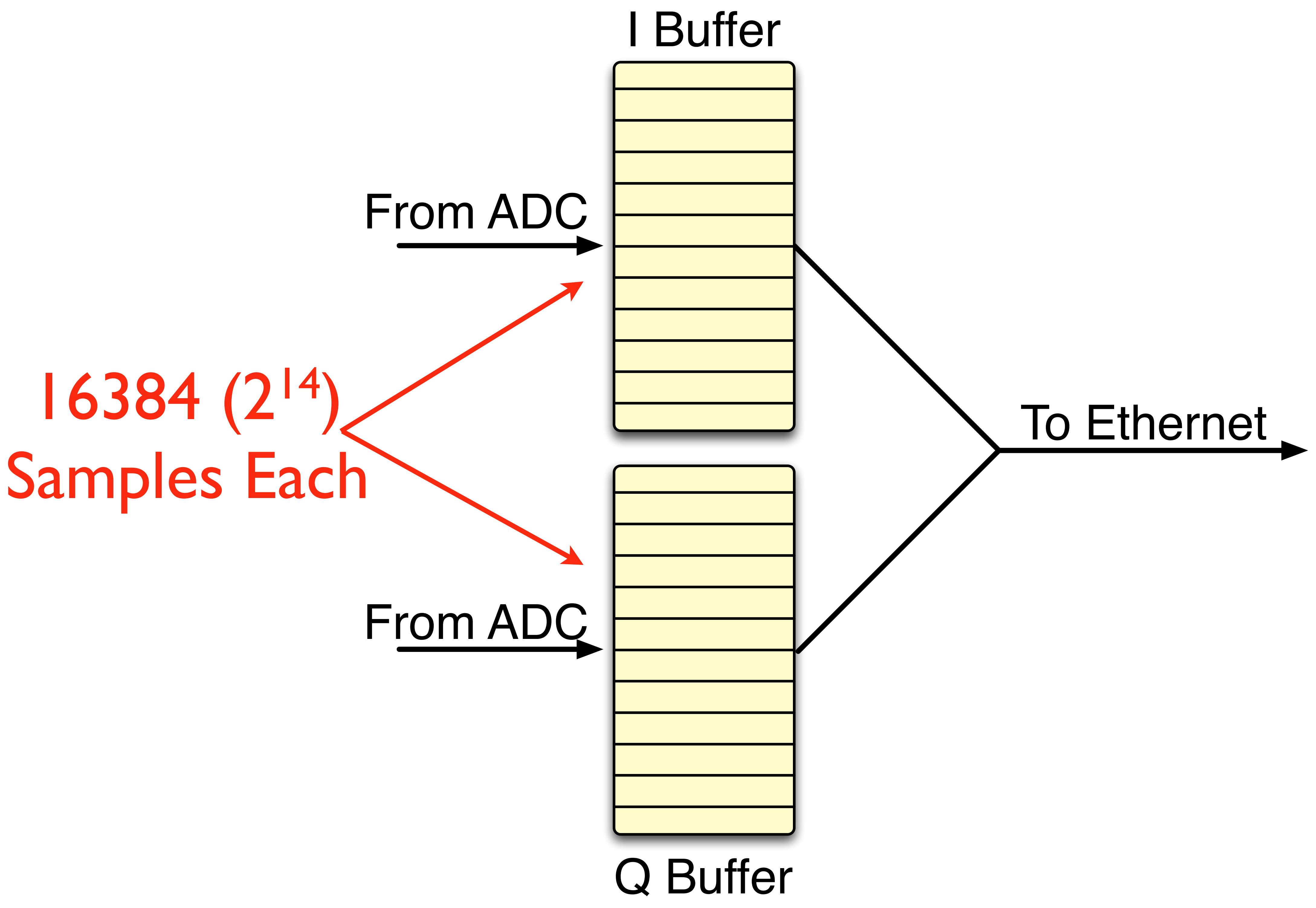


WARPLab Architecture



Transmit Path

WARPLab Architecture



Receive Path

WARPLab Architecture

- Radio control also uses Ethernet
  - Center Frequency
  - Transmit and Receive Gains
- Packet transfers setup various options

WARPLab Flow

1. Initialize nodes & radio settings
2. Download Tx vectors
3. Enable Tx/Rx radio paths
4. Prime Tx/Rx state machines
5. Trigger the transmission and capture
6. Retrieve Rx vectors

WARPLab Examples

- Hardware characterization
- Channel measurement
- Beamforming
- Cooperative communications

Lab 2: Simple Transmitter

- Build a sinusoid generator in Sysgen
- Convert the model to an OPB peripheral
- Connect the Tx core to the radio bridge
- Test the model at RF

Lab 3: WARPLab

- Use WARPLab graphical interface
- Measure the wireless channel
- Build a real bits-to-RF transmitter