Physical Layer Prototyping using WARPLab

Melissa Duarte
Rice University

WARP Workshop
July 14, 2008

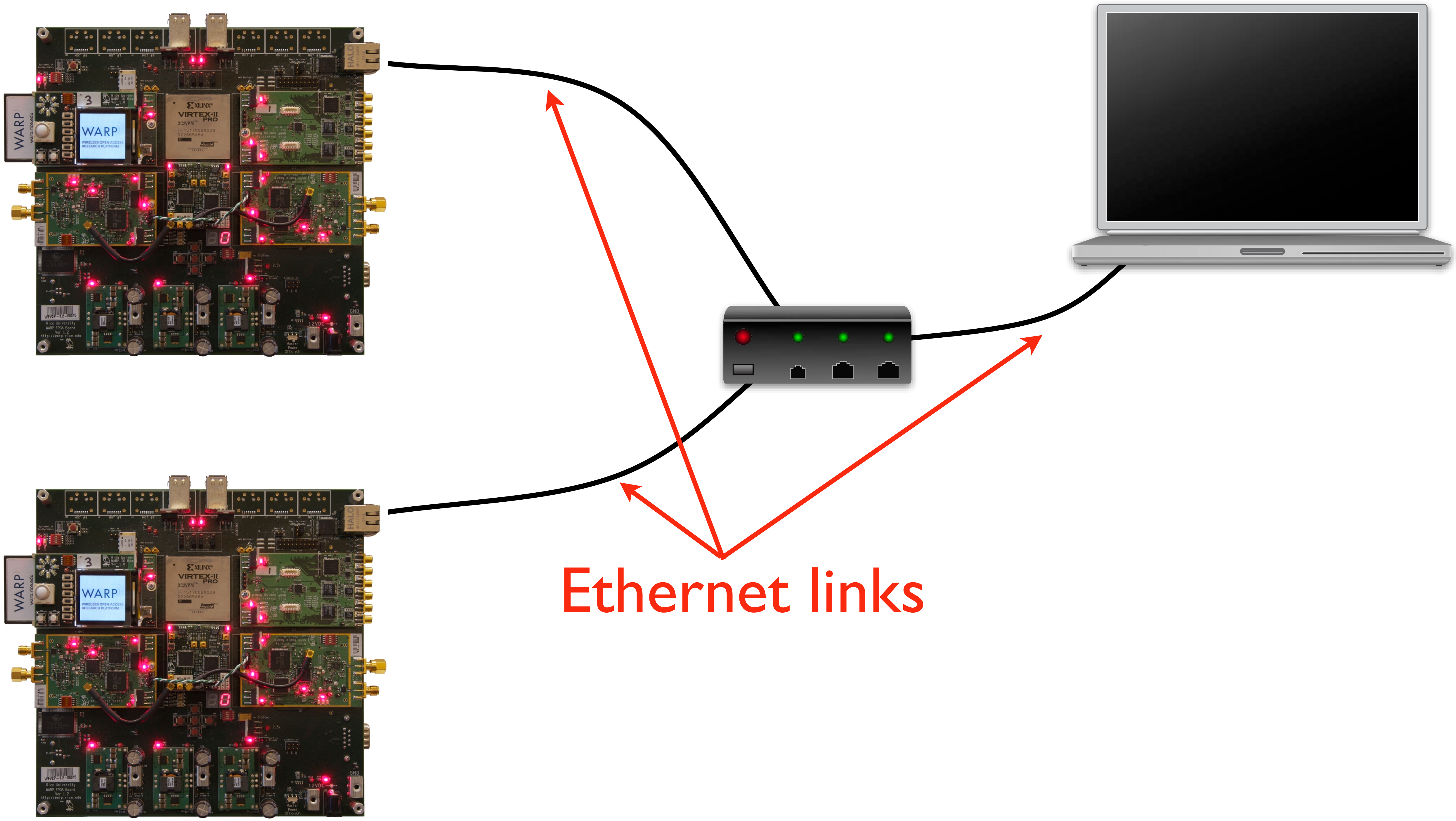
WARP

warp.rice.edu

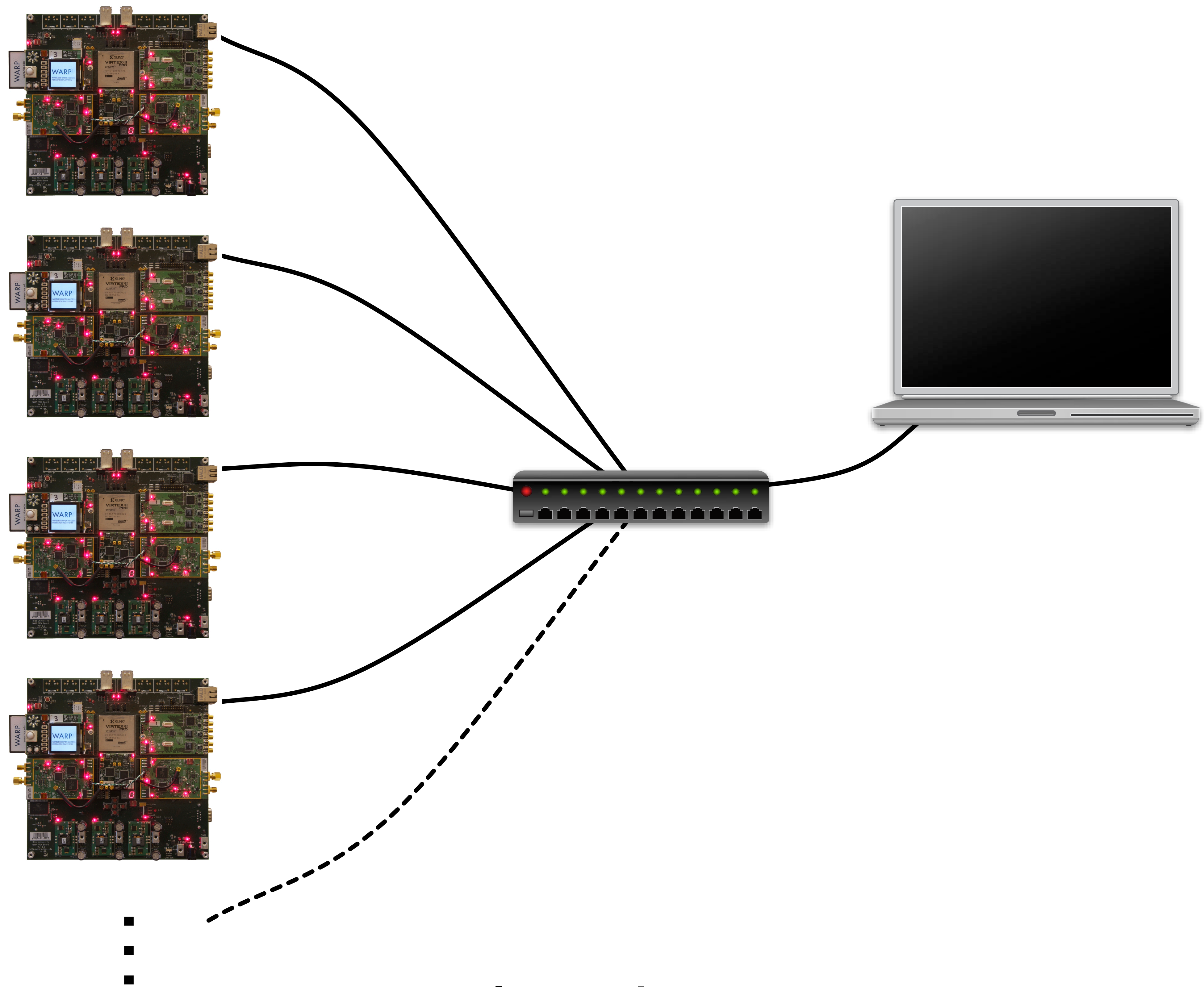
PHY Design Flows

- WARPLab
  - MATLAB↔WARP Link
  - Real-time Tx-Rx and offline processing
  - Very rapid prototyping of PHY algorithms
- Real-time PHY design
  - Low-level FPGA design using Sysgen
  - Putting it all together using XPS

WARPLab Overview

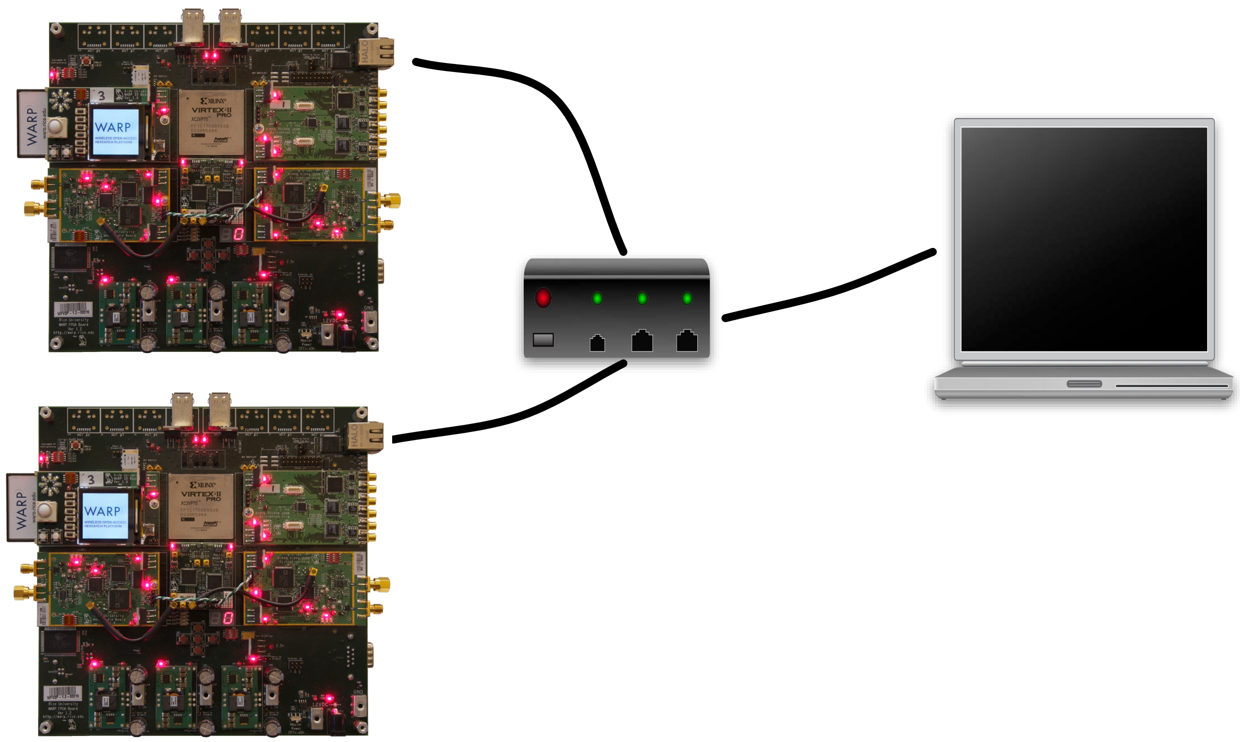


WARPLab Overview



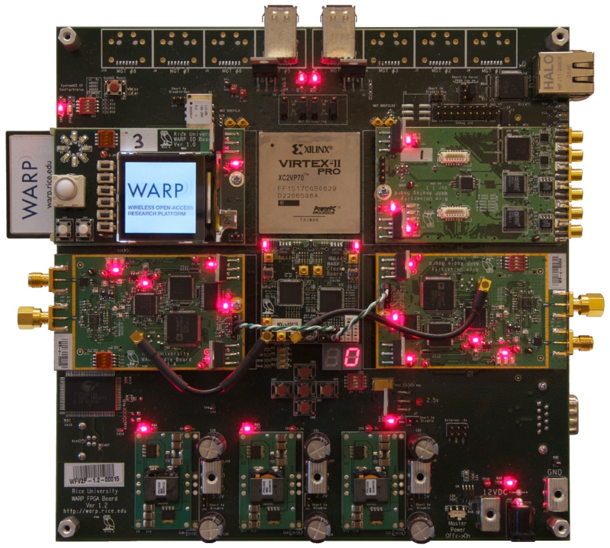
Up to 16 WARP Nodes

WARPLab Overview



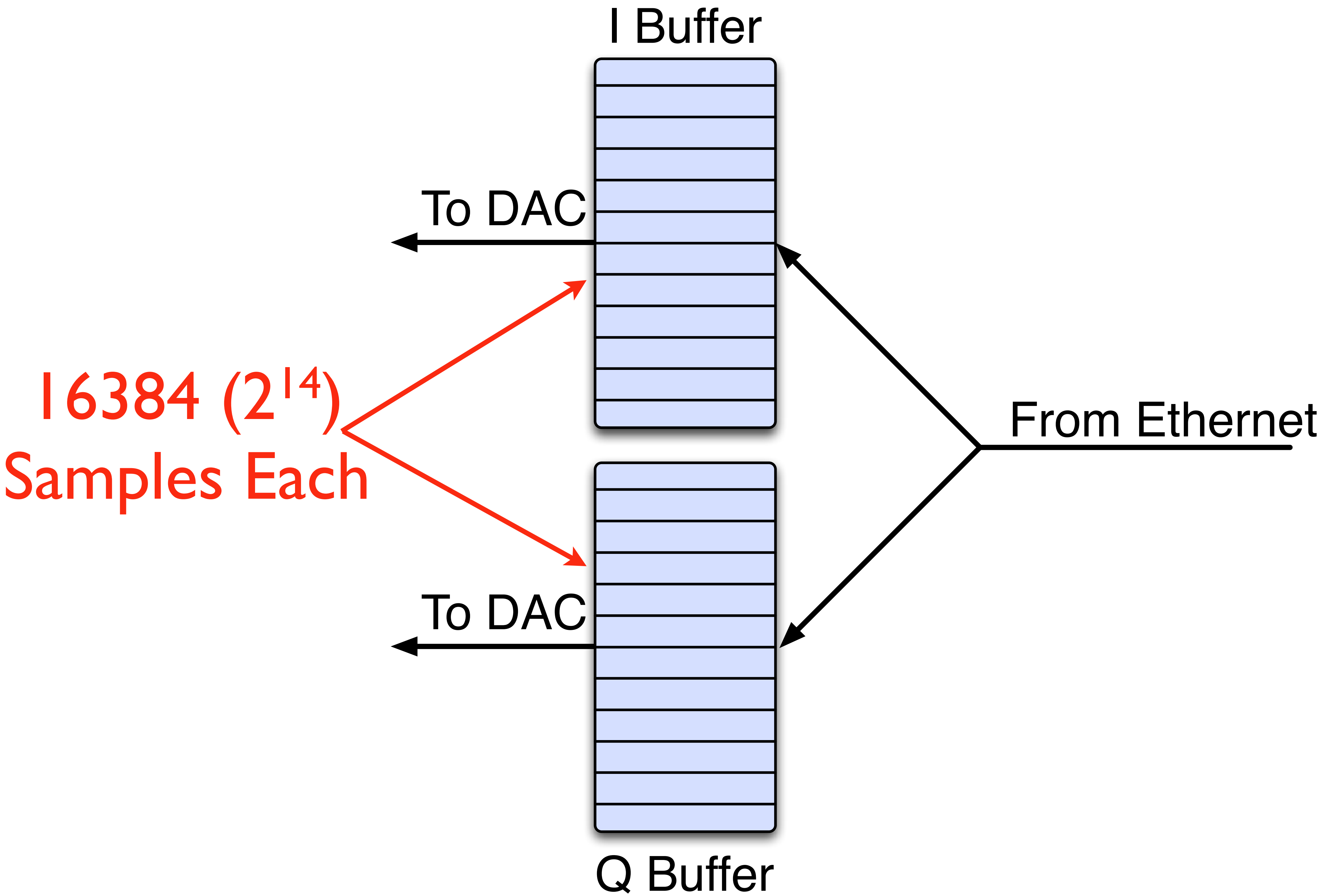
- One PC controls many WARP nodes
- MATLAB for signal processing
- Non-real-time processing
- WARP for wireless interfaces
- Real-time channel use

WARPLab Architecture



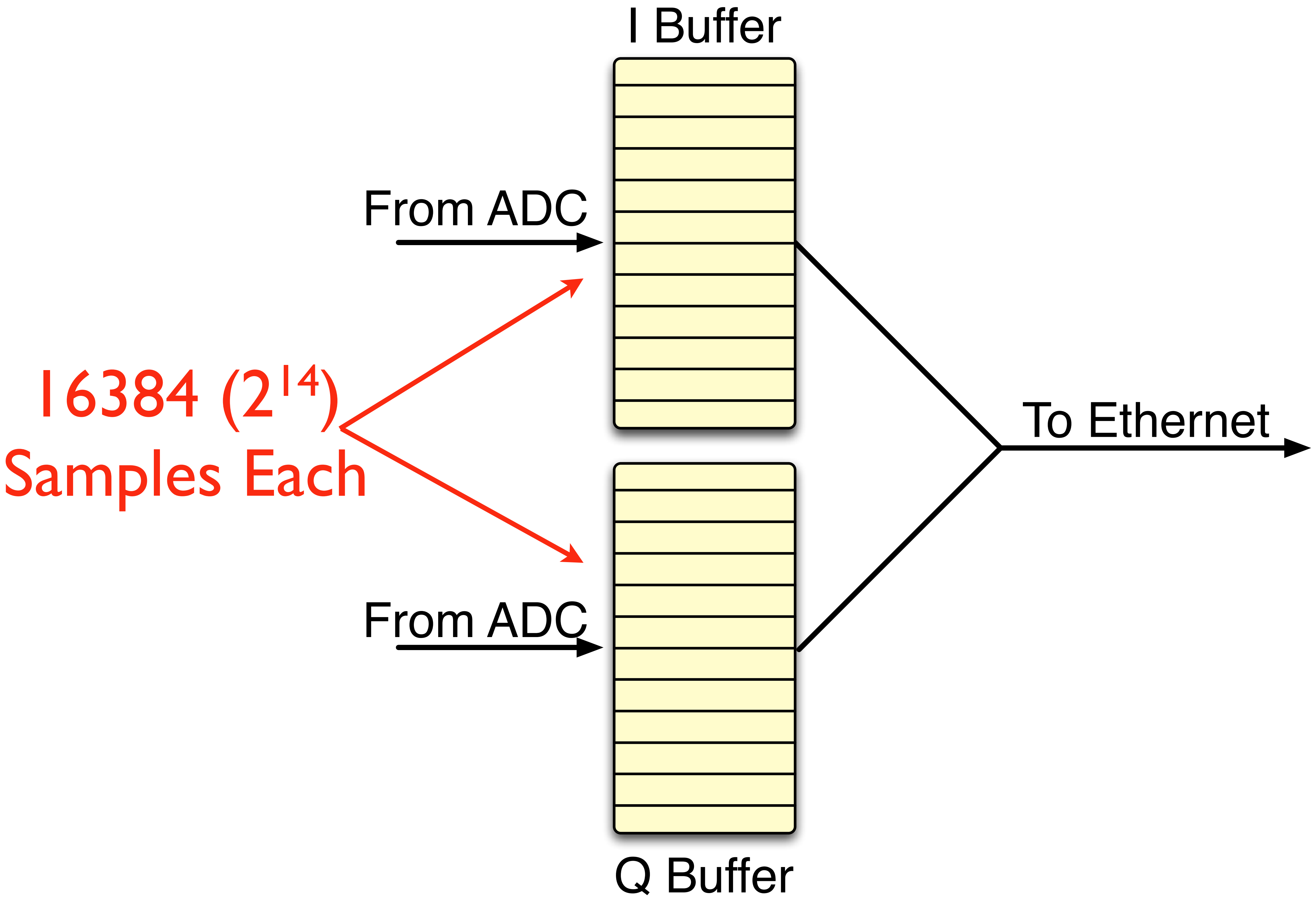
- Bitstream is provided
    - Reference design is provided
- Same bitstream for all nodes
    - Any node can be Tx or Rx
- MATLAB user interface
    - M-code functions provided facilitate interaction with boards
- All open-source

WARPLab Architecture



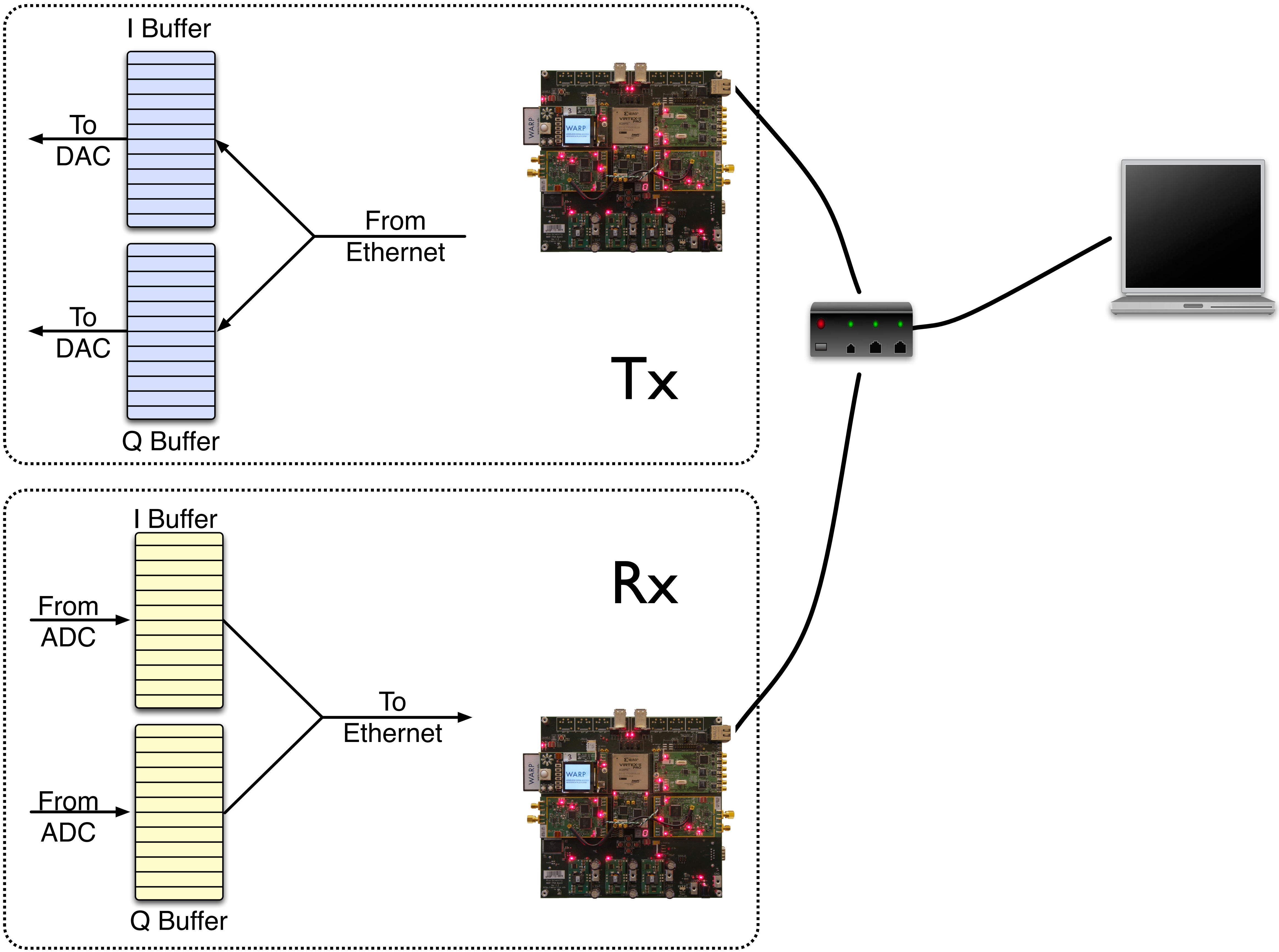
Transmit Path

WARPLab Architecture

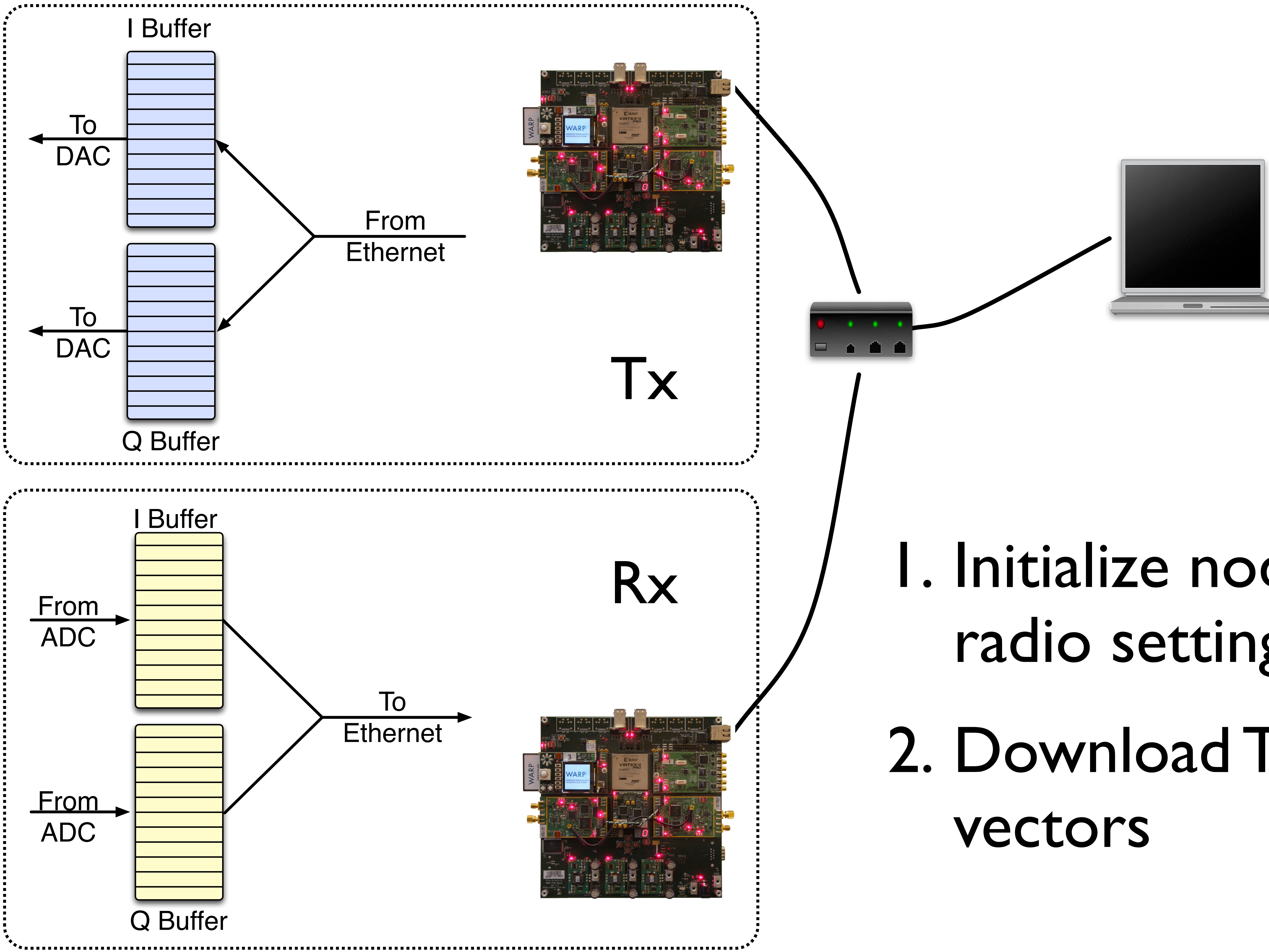


Receive Path

WARPLab Flow

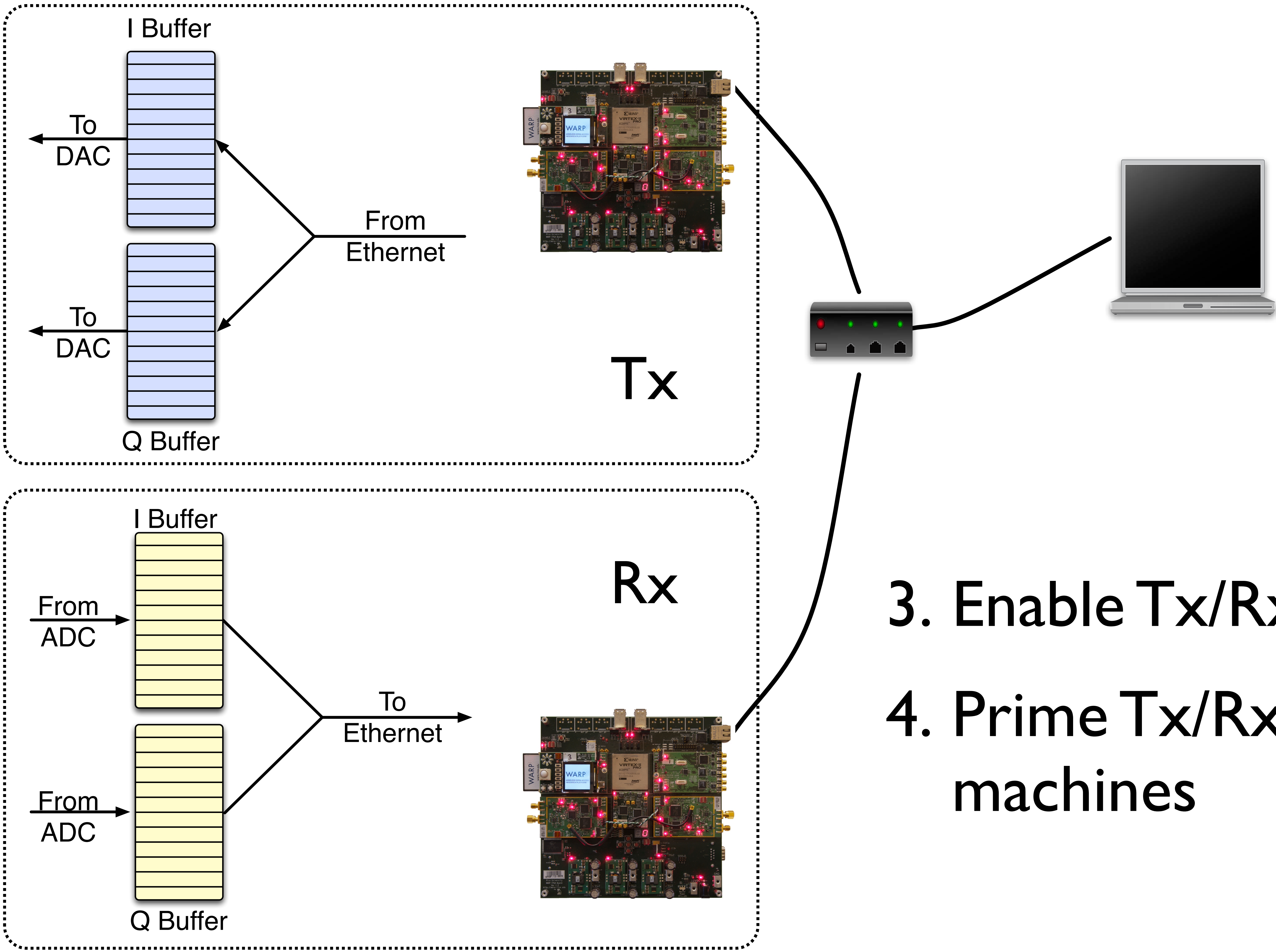


WARPLab Flow



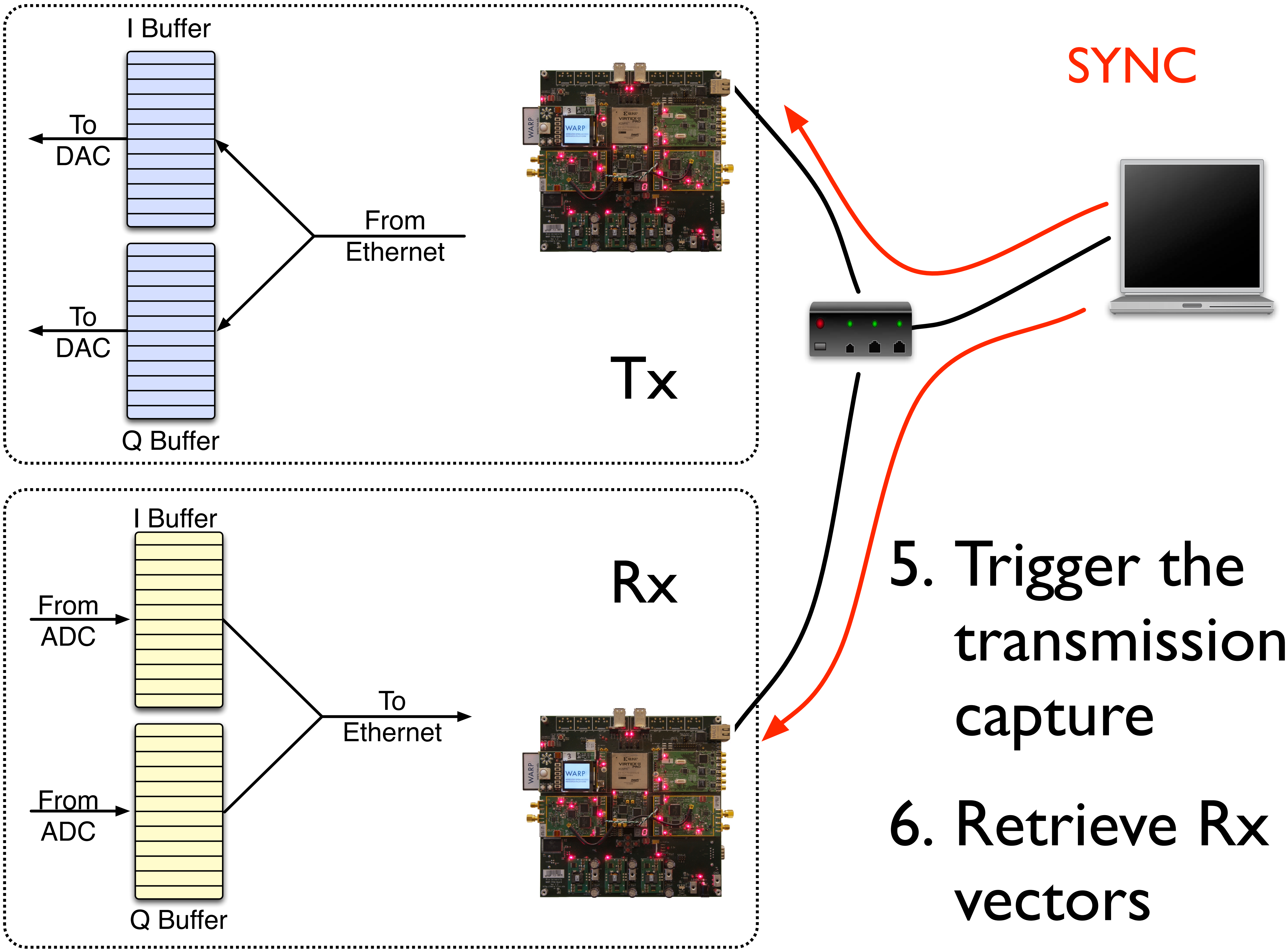
1. Initialize nodes & radio settings
2. Download Tx vectors

WARPLab Flow



3. Enable Tx/Rx paths
4. Prime Tx/Rx state machines

WARPLab Flow

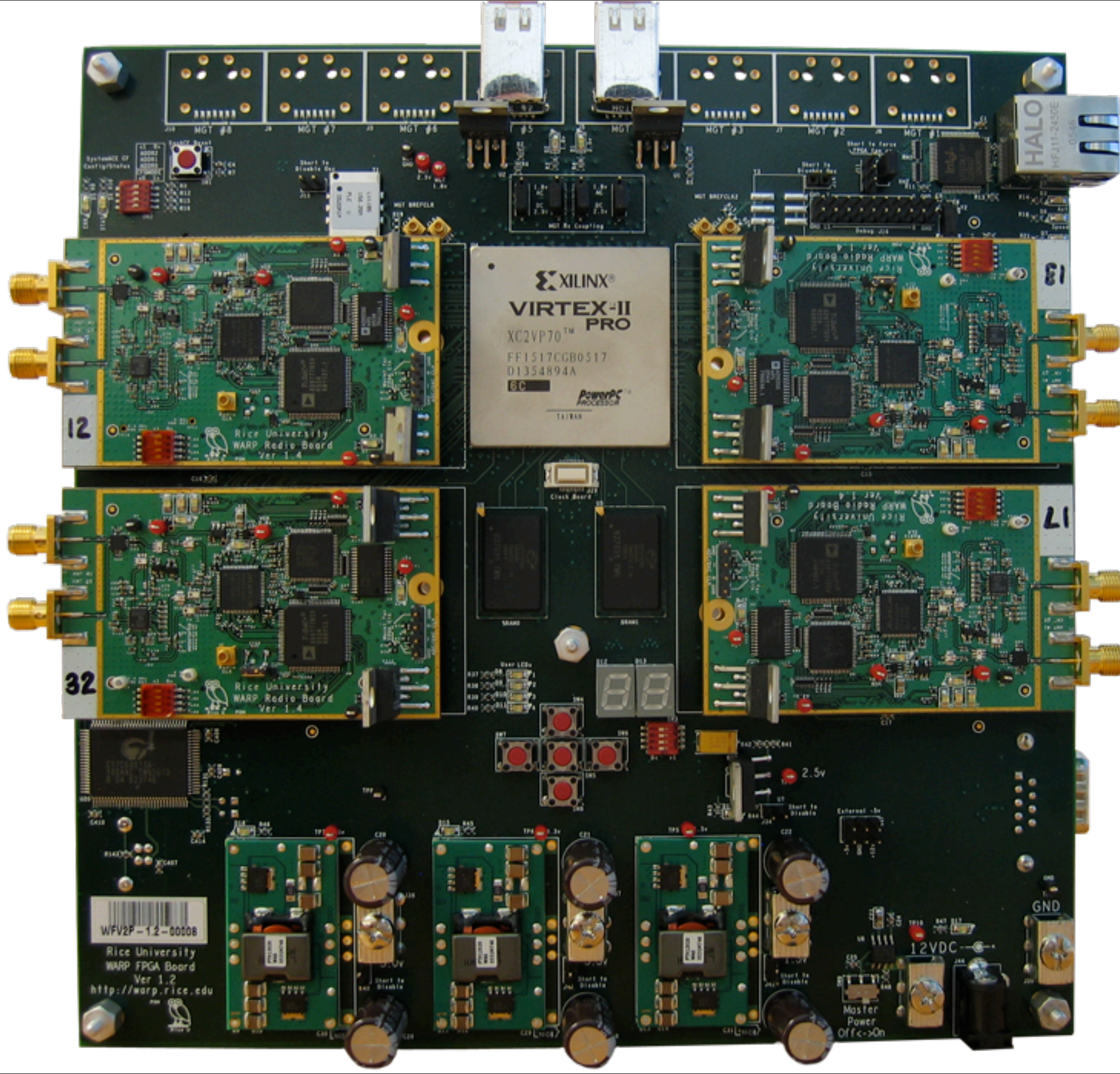


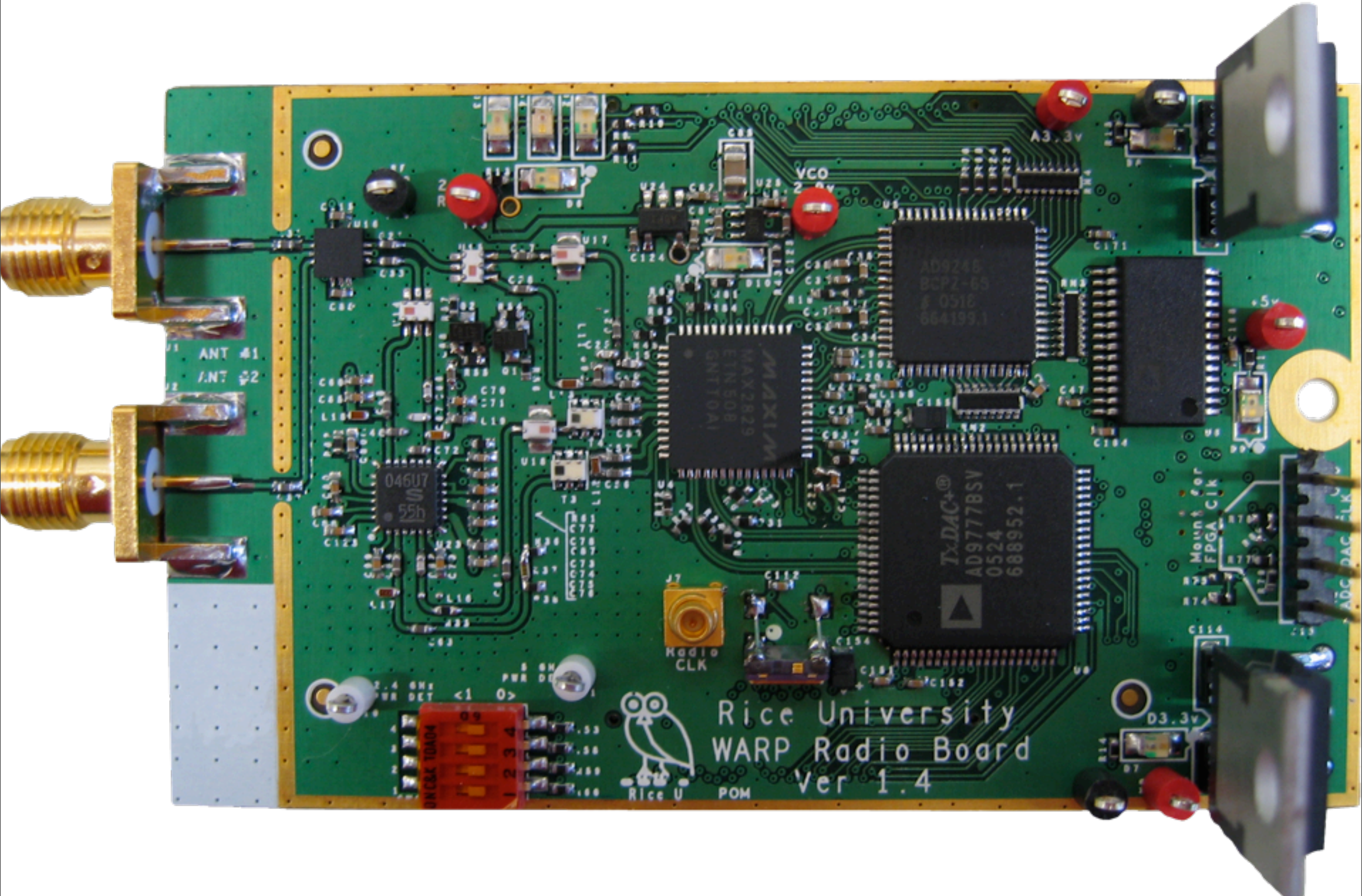
5. Trigger the transmission and capture

6. Retrieve Rx vectors

WARPLab Architecture

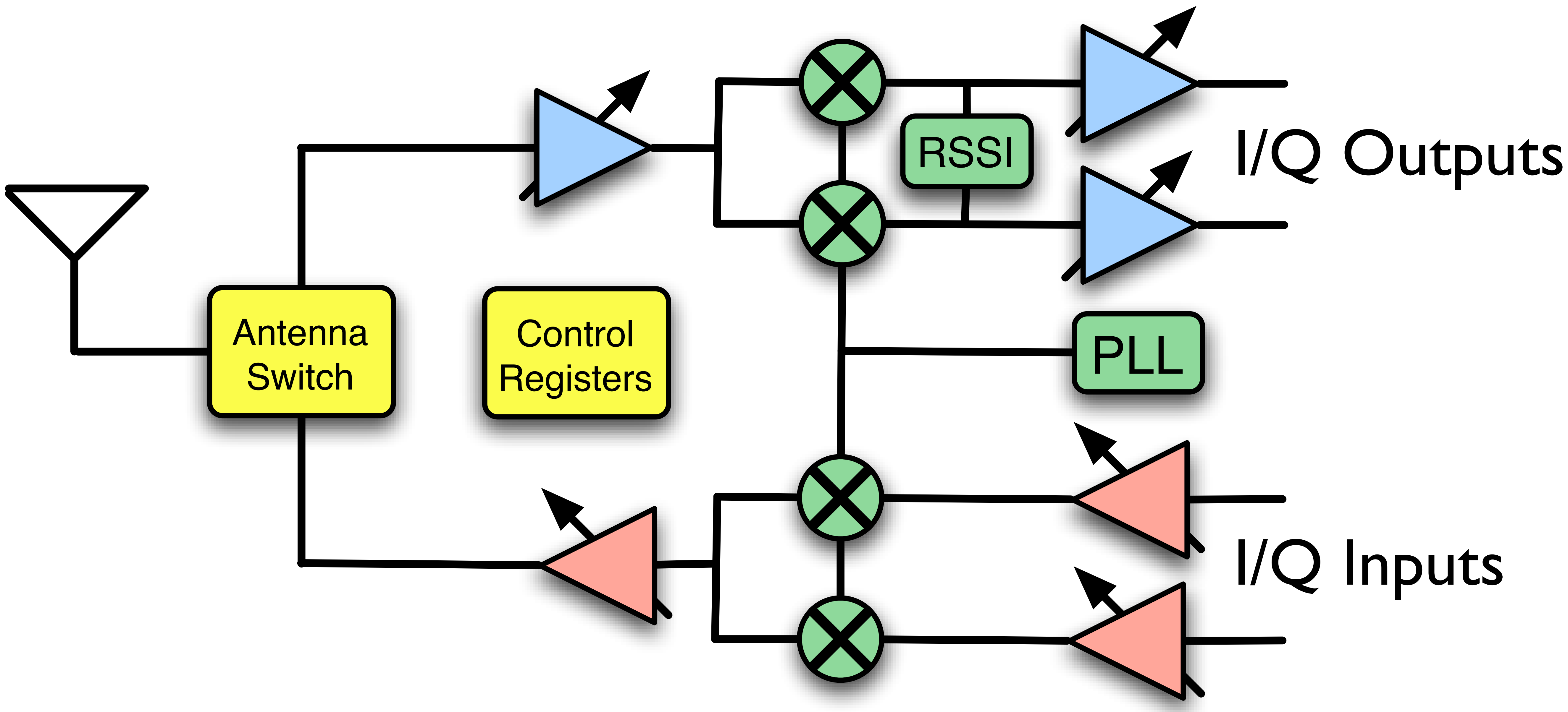
- Independent Tx/Rx buffers
  - Buffers persist between triggers
  - Rx path captures I/Q and RSSI
- Control and synchronization by Ethernet
- Low-level radio control
  - Tx/Rx gains
  - Center frequency


XILINX
VIRTEX-II PRO
XC2VP70
FF1517CGB0517
D1354894A
6C
PowerPC PROCESSOR
HALO
Rice University
WARP Radio Board
Ver 1.4
12
32
Rice University
WARP FPGA Board
Ver 1.2
http://warp.rice.edu
WFV2P-1.2-00008
2.5v
12VDC
GND
Master
Power
Off<->On

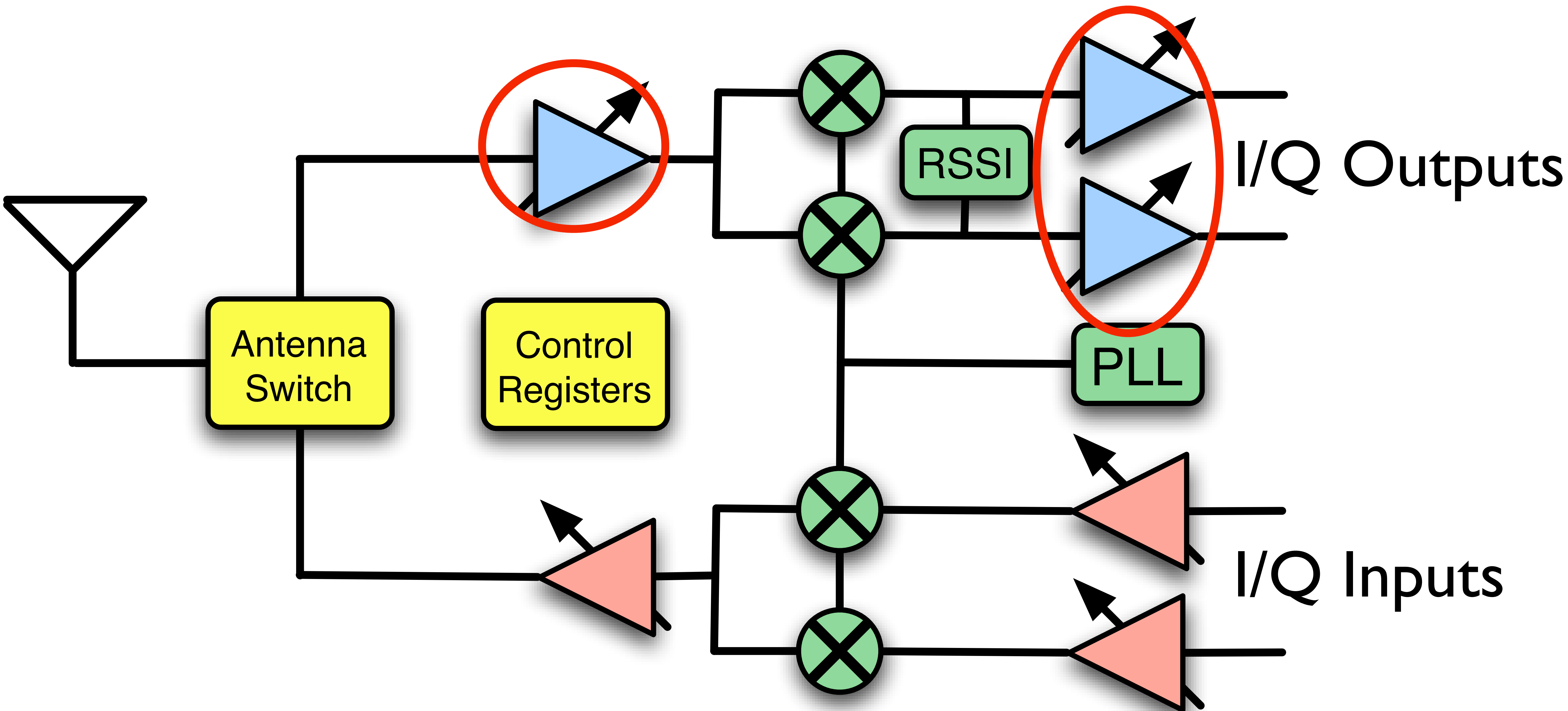


Rice University
WARP Radio Board
Ver 1.4
Rice U
POM
MAXIM
MAX2829
AD9248
BCPZ-65
TxDAC+
AD9777BSV
Radio
CLK
ANT #1
ANT #2
A3.3v
VCO
+5v
D3.3v
Mount for
FPGA Clk

Radio Transceiver

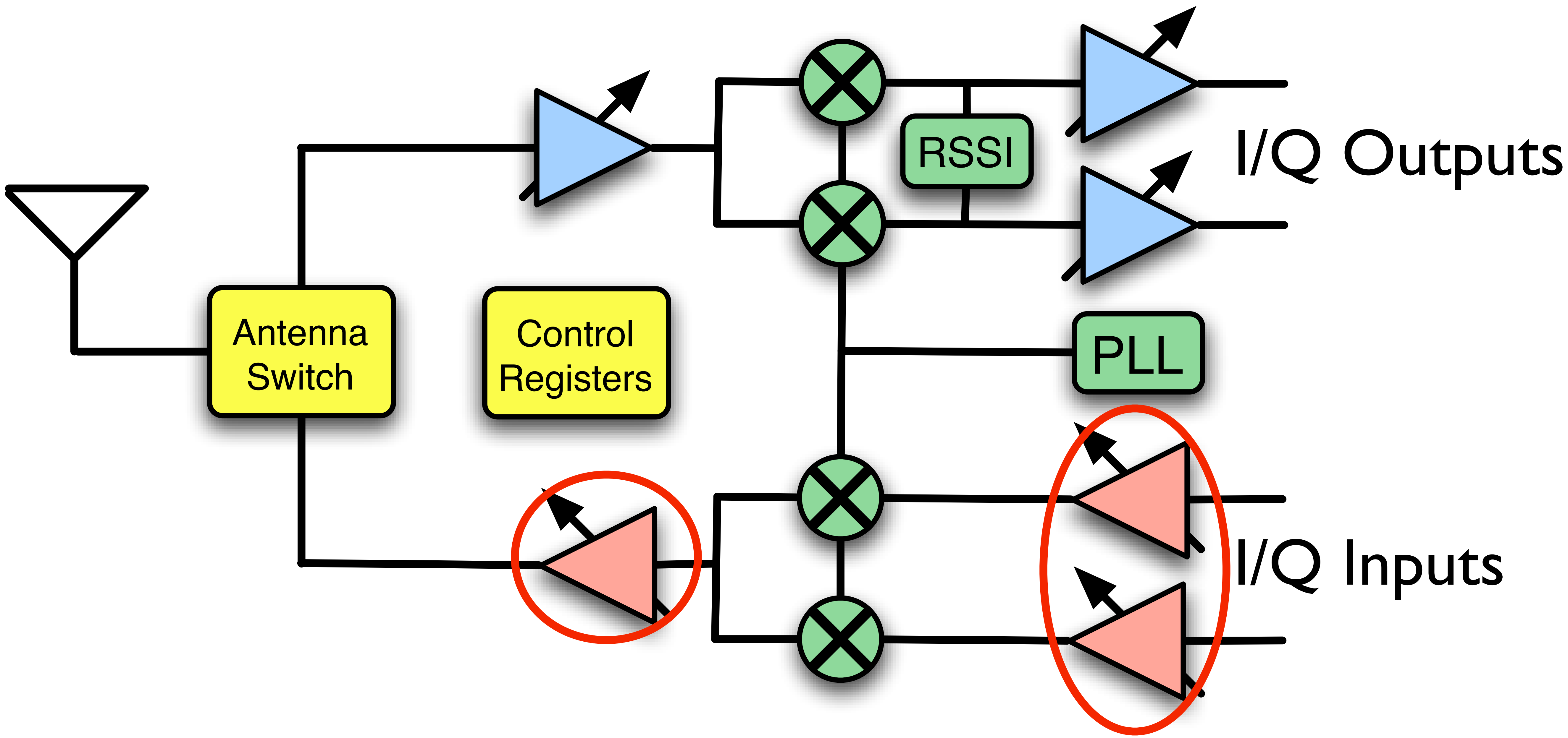


Radio Transceiver



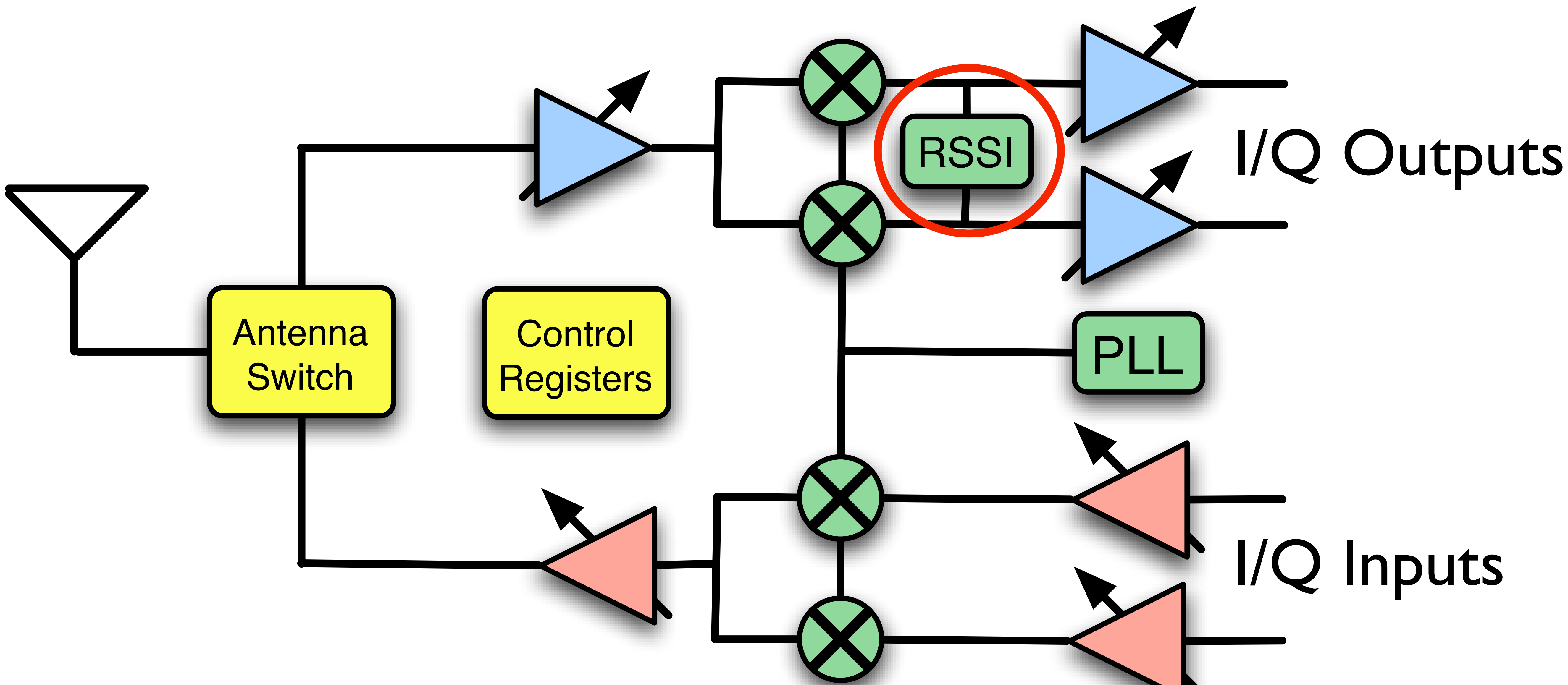
Variable Gain Rx Amplifiers
Gain value input from MATLAB

Radio Transceiver



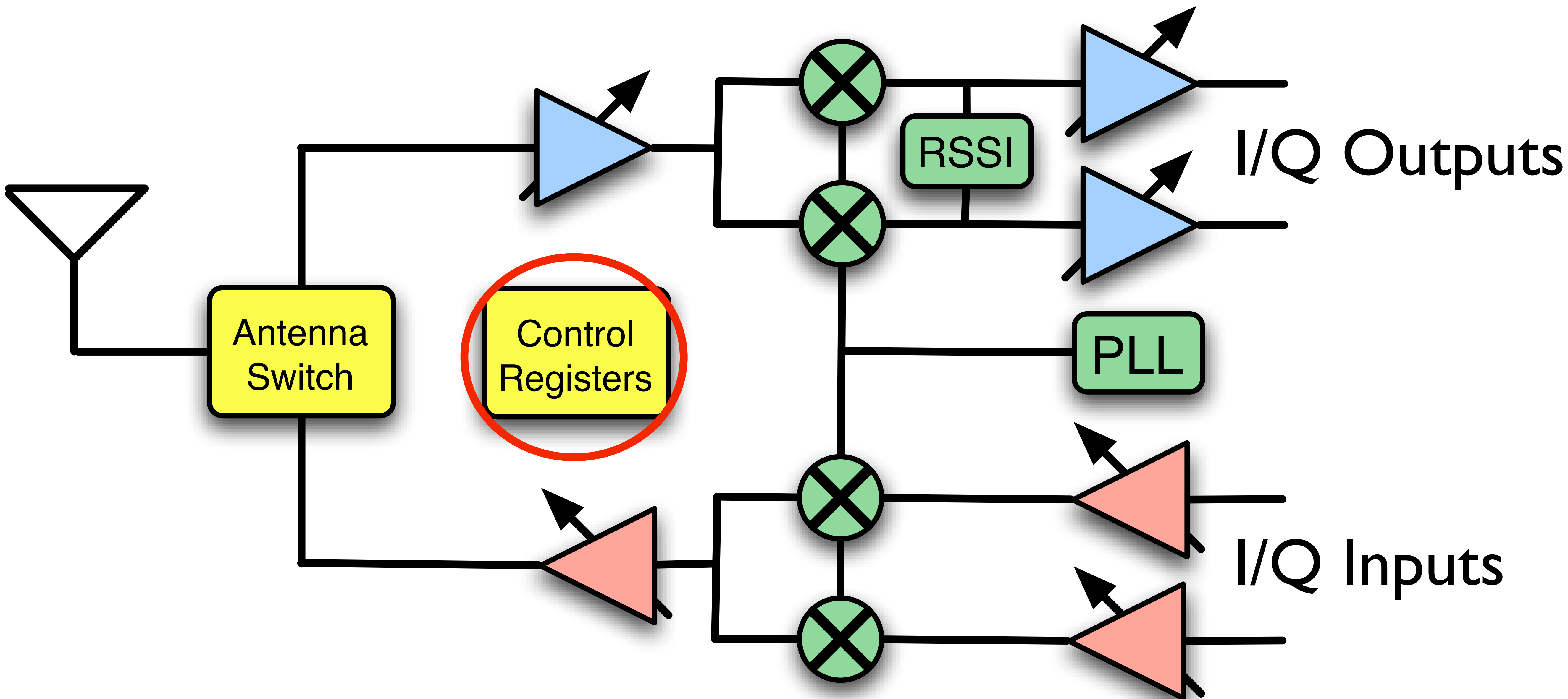
Variable Gain Tx Amplifiers
Gain value input from MATLAB

Radio Transceiver



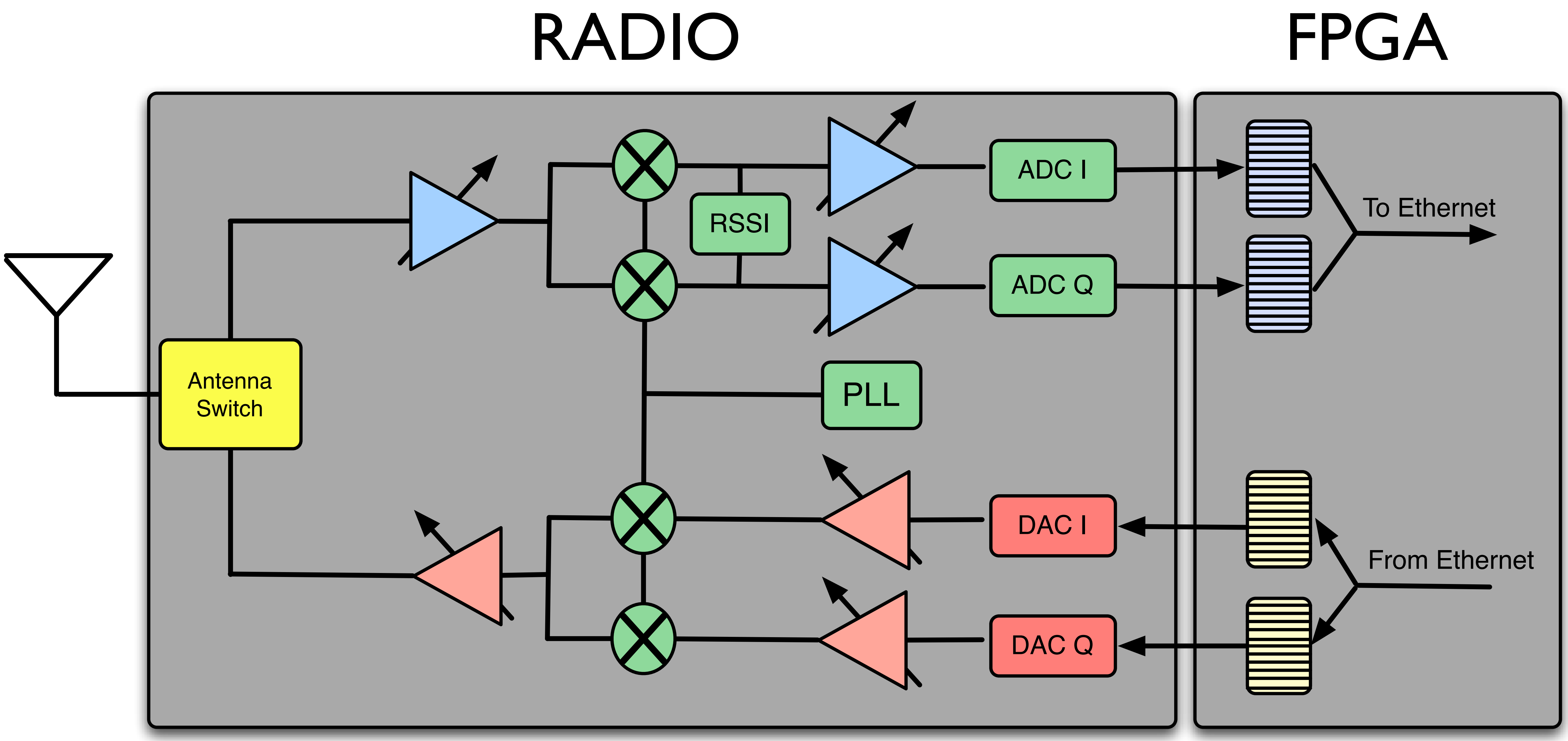
Received Signal Strength Indicator After RF Gain
Written in a buffer in the FPGA
Can be read from MATLAB

Radio Transceiver



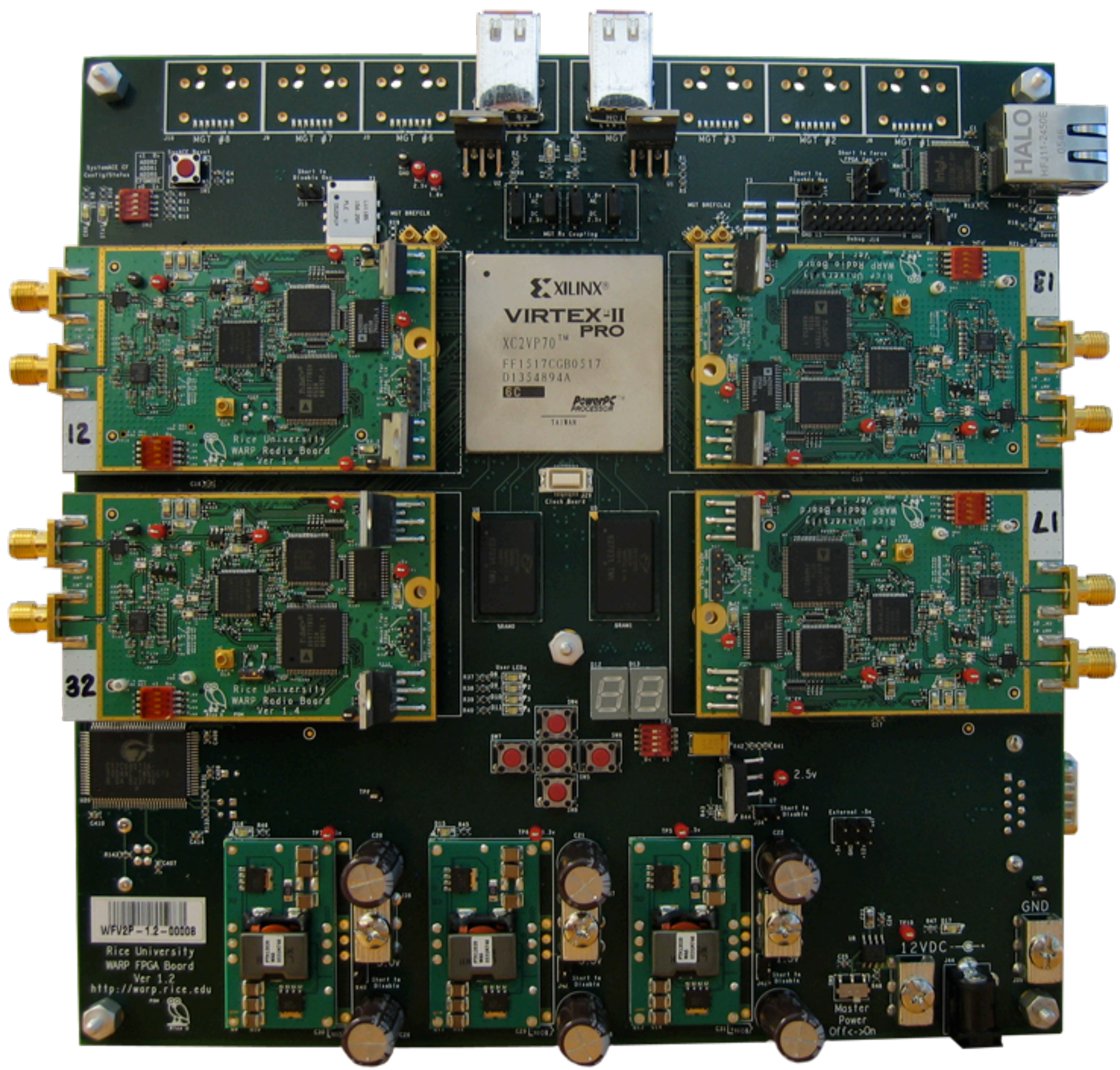
Register Bank
(controlled by SPI interface)

WARPLab Architecture



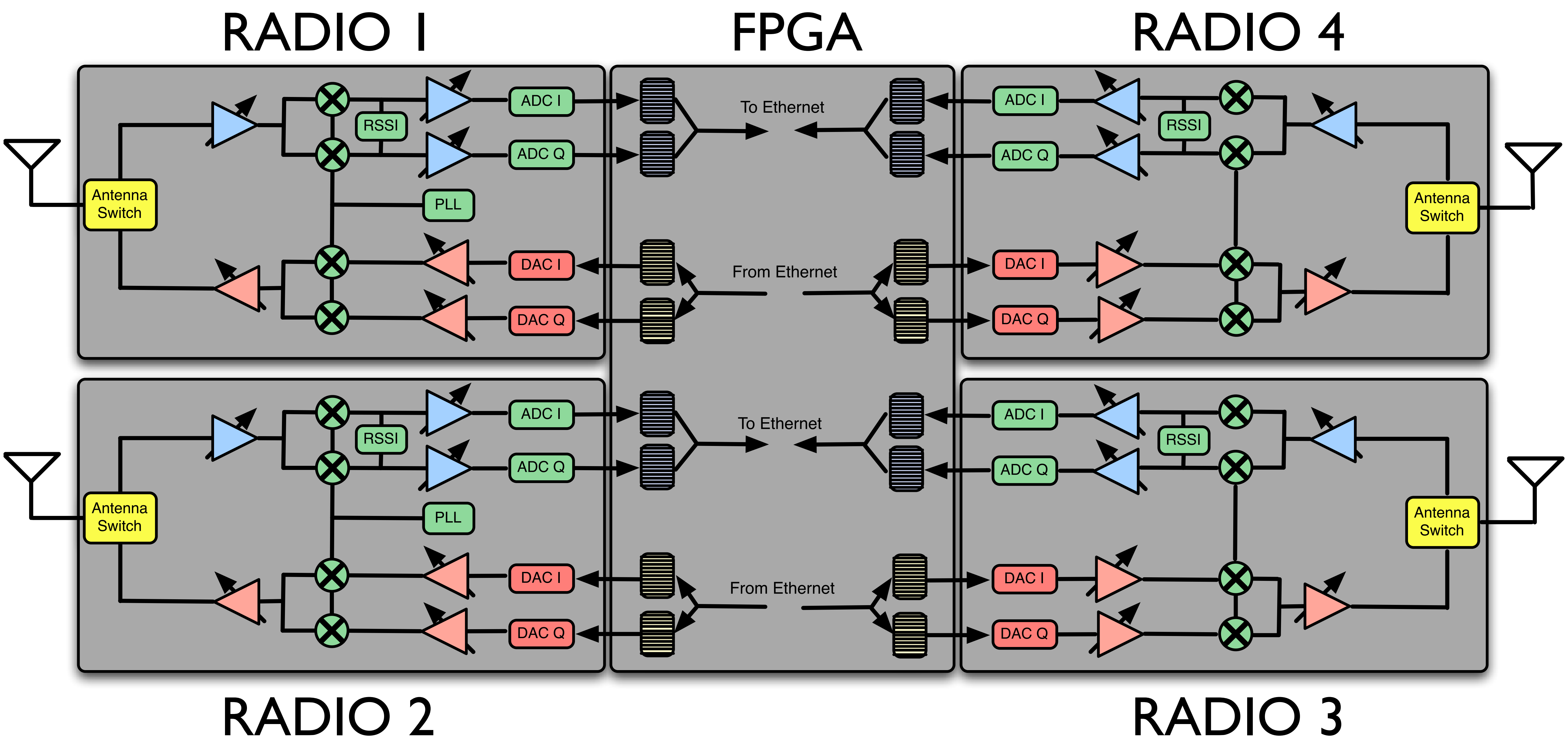
WARPLab Architecture

4x4 node



WARPLab Architecture

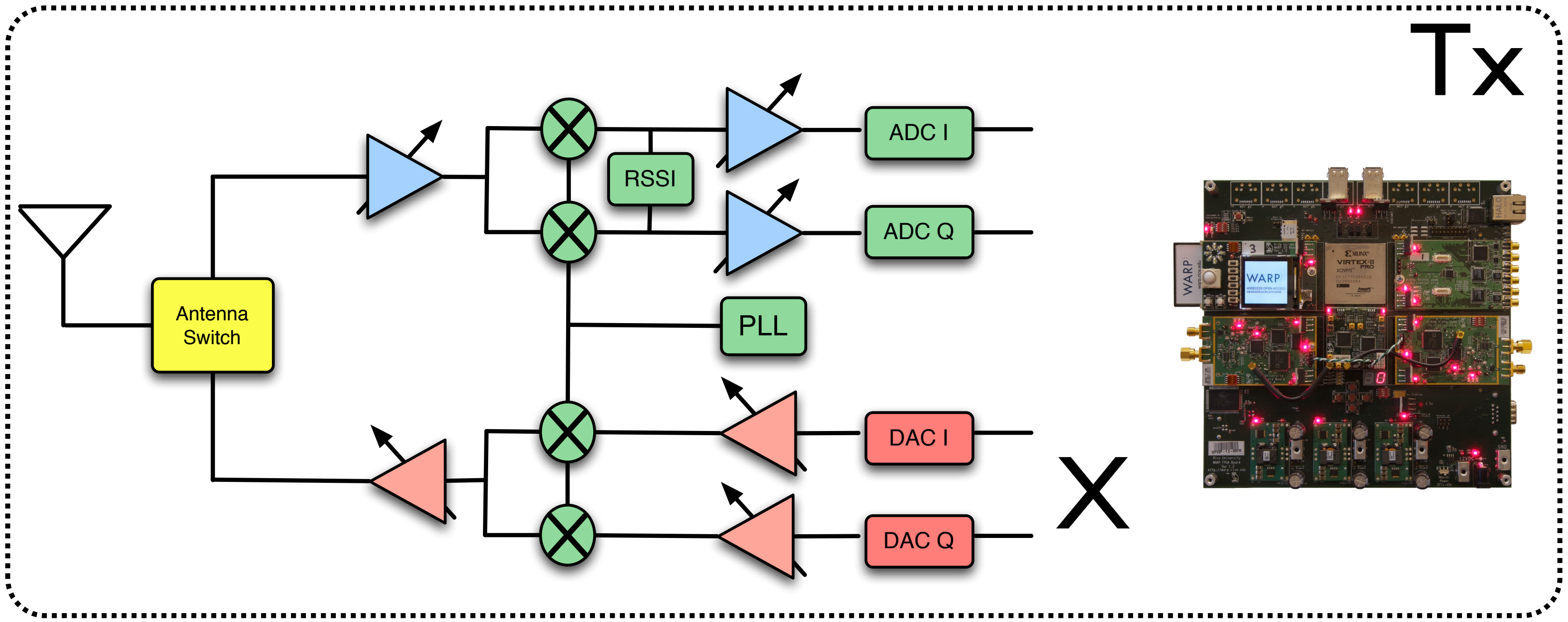
4x4 node



Tx Signal Requirements

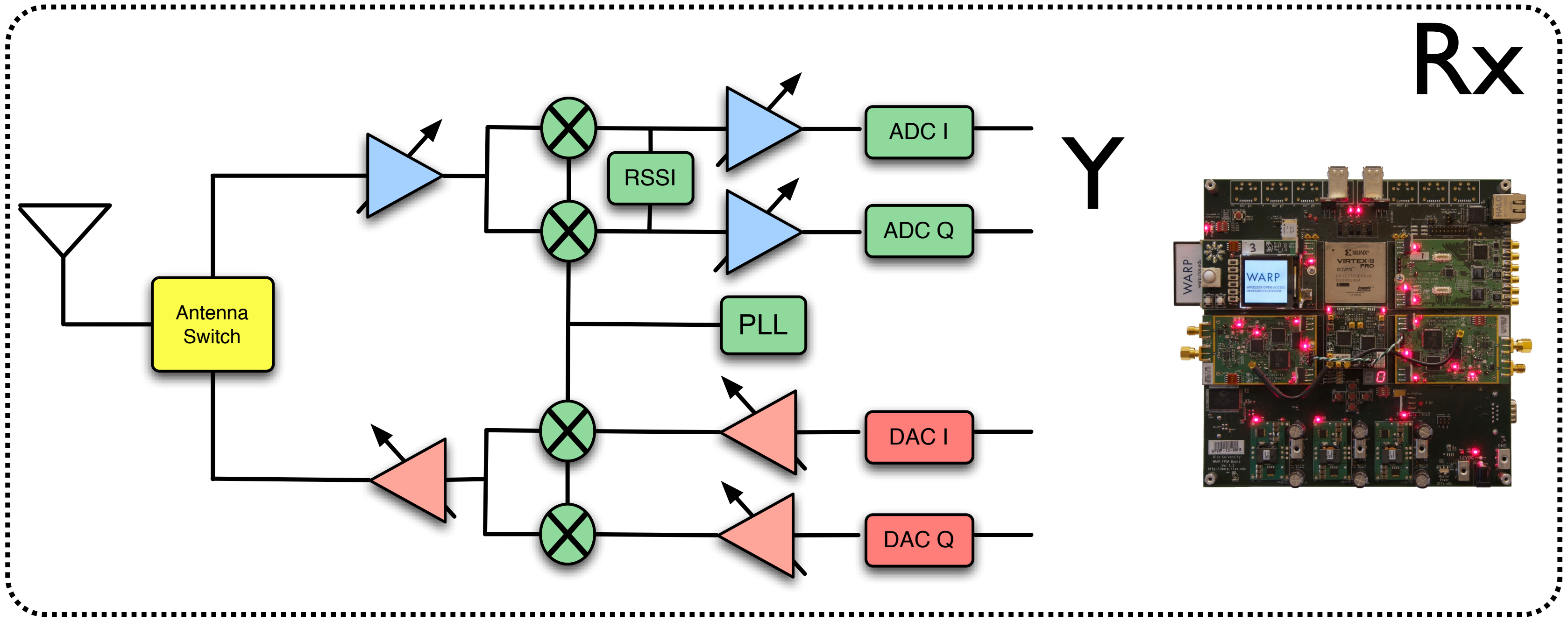
- Amplitude of real part in [-1,1]
- Amplitude of imaginary part in [-1,1]
- Highest frequency 9.5 MHz (19 MHz BW)
- Lowest frequency 30 KHz
- 40 MHz sampling

Tx to Rx path



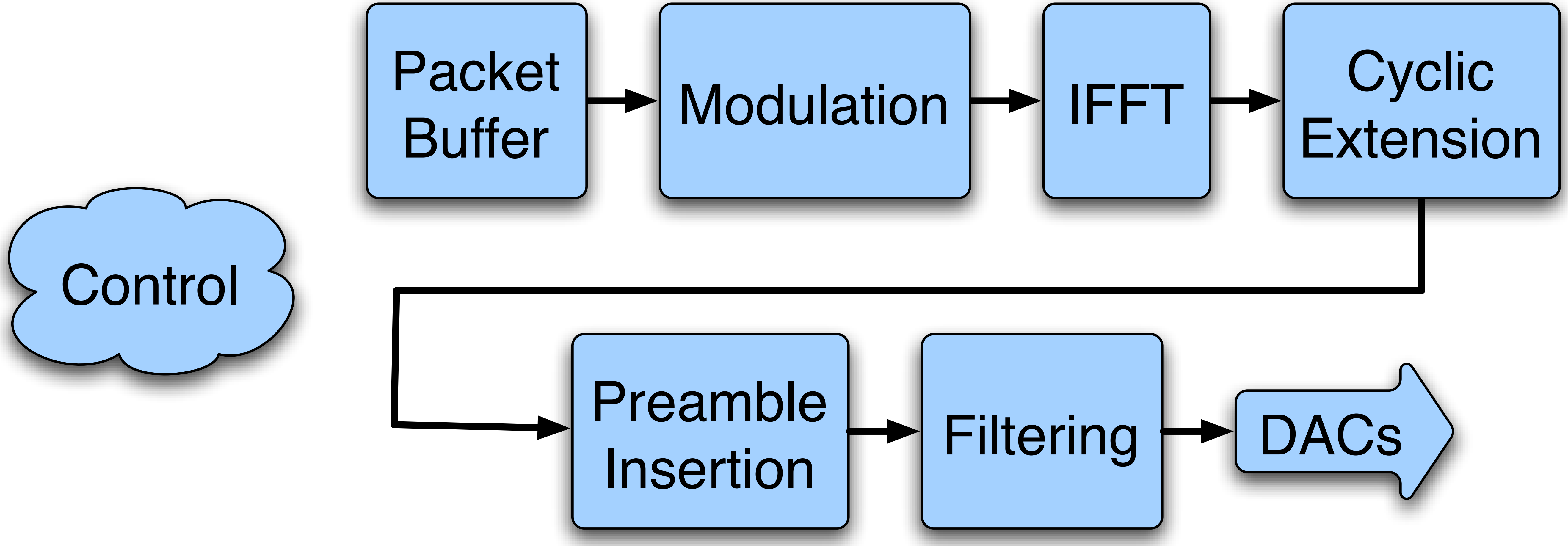
$Y = |H| \, G_{RXBB} \, G_{RXRF} \, G_{TXPA} \, G_{TXRF} \, G_{TXBB} \, X$

|H| : Wireless channel magnitude



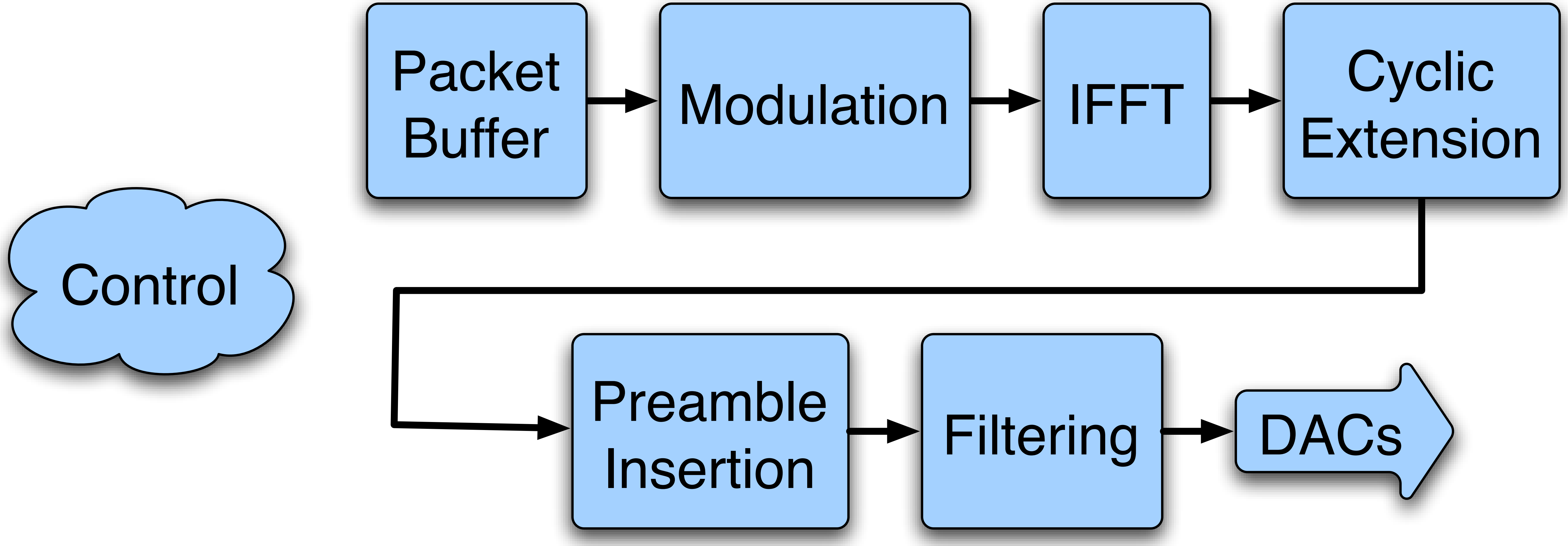
PHY Example: OFDM Tx

WARPLab offline vs. Sysgen real-time



PHY Example: OFDM Tx

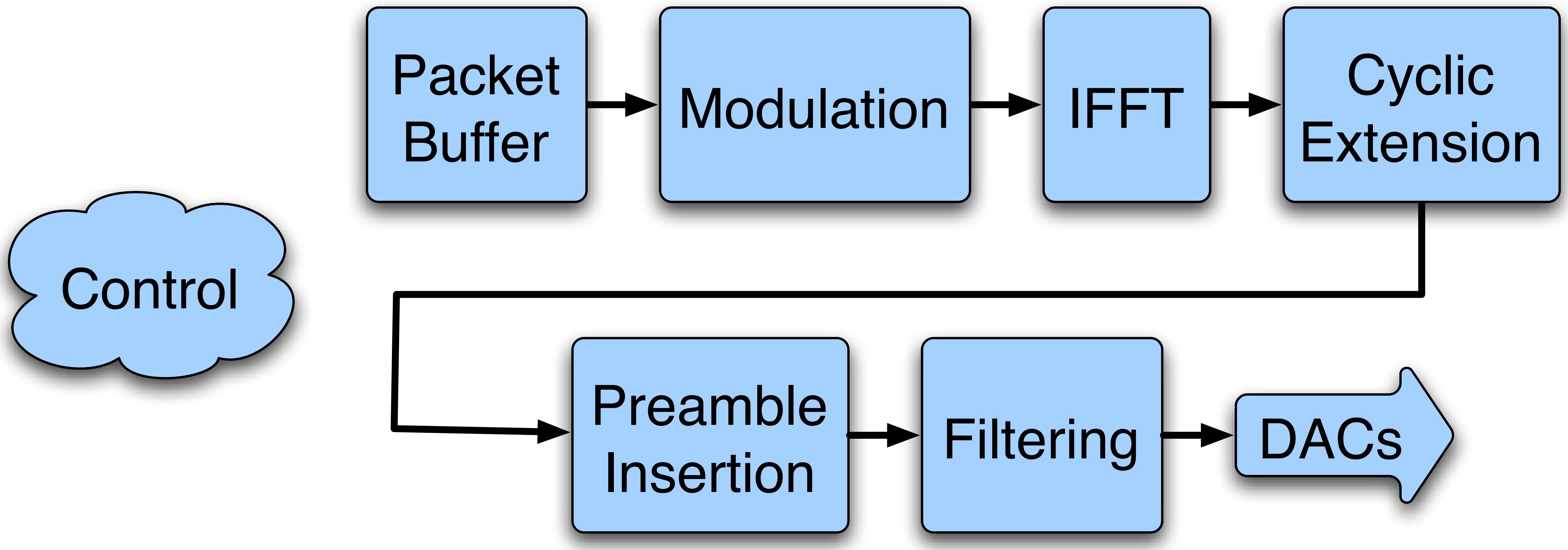
WARPLab offline vs. Sysgen real-time



- Offline Matlab processing
- Download samples to board
- Real-time Tx/Rx

PHY Example: OFDM Tx

WARPLab offline vs. Sysgen real-time



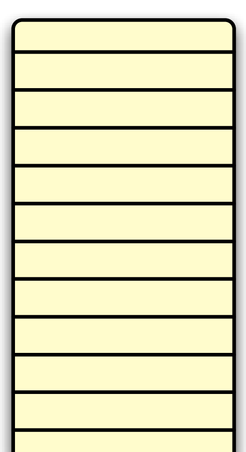
- Each sample to DACs is generated in real-time
- Real-time Tx/Rx

PHY Example: OFDM Tx

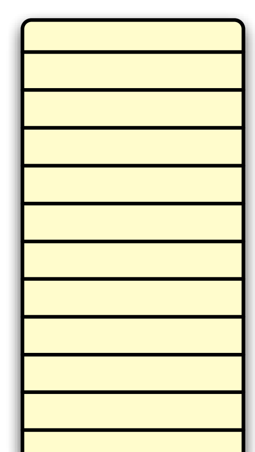
WARPLab offline vs. Sysgen real-time

```
trainingSym_freq = [0 -1 1 -1 1 -1 1 -1 1 -1 -1 -1 -1 1 1 -1 1 1 1 1 1 -1 -1 1 1 -1 -1 0 0 0...
 0 0 0 0 0 0 0 0 1 -1 1 1 1 -1 -1 -1 -1 1 1 -1 1 -1 1 1 -1 1 1 1 -1 -1 -1 -1 -1 -1];
trainingSyms_time = ifft(trainingSym_freq, 64);
numBytes = 240;
numOFDMSyms = 20;
fftWindowOffset = 6;
numTrainingSyms = 4;
pilotTonesTx = (randint(numOFDMSyms, 4, 2) .* 2) - 1;

dataTx_qpsk = randint(1, numBytes*4, 4);
mod_qpsk = qammod([0:3], 4, 0, 'gray');
modnorm_qpsk = modnorm(mod_qpsk, 'avpow', 1);
symsTx_qpsk = repmat([zeros(1,47) 1], 1, 20);%qammod(dataTx_qpsk, 4, 0, 'gray') .* modnorm_qpsk;
symsTx_qpsk = qammod(dataTx_qpsk, 4, 0, 'gray') .* modnorm_qpsk;
symsTx_mat = reshape(symsTx_qpsk, 48, numOFDMSyms).'; %20x48
ifftInTx = zeros(20, 64);

%Start filling in the FFT input matrix, leaving zeros where needed
% Columns: 1->DC, 32->max +freq, 33->max -freq, 64->min -freq
% Leave zeros in columns([1 8 22 28:32 33:38 44 58])
ifftInTx(:, [2:7 9:21 23:27 39:43 45:57 59:64]) = symsTx_mat; %20x64
ifftInTx(:, [8 22 44 58]) = pilotTonesTx;
ifftInTx = [repmat(trainingSym_freq, numTrainingSyms, 1); ifftInTx];
ifftOutTx = ifft(ifftInTx.', 64).';

cycExtTx = [ifftOutTx(:,49:64) ifftOutTx];
TxVec = reshape(cycExtTx.', 1, prod(size(cycExtTx)));

TxVec_air = [zeros(1,250) interp(TxVec, 4) zeros(1,2^14 - length(TxVec)*4 - 250)];
```

Final Output to DACs



I Buffer



Q Buffer

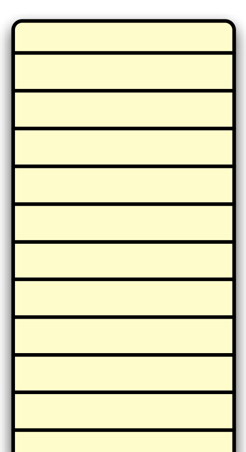
PHY Example: OFDM Tx

WARPLab offline vs. Sysgen real-time

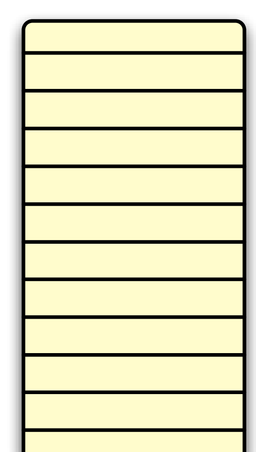
```
trainingSym_freq = [0 -1 1 -1 1 -1 1 -1 1 -1 -1 -1 -1 1 1 -1 1 1 1 1 1 -1 -1 1 1 -1 -1 0 0 0...
 0 0 0 0 0 0 0 0 1 -1 1 1 1 -1 -1 -1 -1 1 1 -1 1 -1 1 1 -1 1 1 1 -1 -1 -1 -1 -1 -1];
trainingSyms_time = ifft(trainingSym_freq, 64);
numBytes = 240;
numOFDMSyms = 20;
fftWindowOffset = 6;
numTrainingSyms = 4;
pilotTonesTx = (randint(numOFDMSyms, 4, 2) .* 2) - 1;

dataTx_qpsk = randint(1, numBytes*4, 4);
mod_qpsk = qammod([0:3], 4, 0, 'gray');
modnorm_qpsk = modnorm(mod_qpsk, 'avpow', 1);
symsTx_qpsk = repmat([zeros(1,47) 1], 1, 20);%qammod(dataTx_qpsk, 4, 0, 'gray') .* modnorm_qpsk;
symsTx_qpsk = qammod(dataTx_qpsk, 4, 0, 'gray') .* modnorm_qpsk;
symsTx_mat = reshape(symsTx_qpsk, 48, numOFDMSyms).'; %20x48
ifftInTx = zeros(20, 64);

%Start filling in the FFT input matrix, leaving zeros where needed
% Columns: 1->DC, 32->max +freq, 33->max -freq, 64->min -freq
% Leave zeros in columns([1 8 22 28:32 33:38 44 58])
ifftInTx(:, [2:7 9:21 23:27 39:43 45:57 59:64]) = symsTx_mat; %20x64
ifftInTx(:, [8 22 44 58]) = pilotTonesTx;
ifftInTx = [repmat(trainingSym_freq, numTrainingSyms, 1); ifftInTx];
ifftOutTx = ifft(ifftInTx.', 64).';

cycExtTx = [ifftOutTx(:,49:64) ifftOutTx];
TxVec = reshape(cycExtTx.', 1, prod(size(cycExtTx)));

TxVec_air = [zeros(1,250) interp(TxVec, 4) zeros(1,2^14 - length(TxVec)*4 - 250)];
```

Packet generation and modulation

Final Output to DACs



I Buffer



Q Buffer

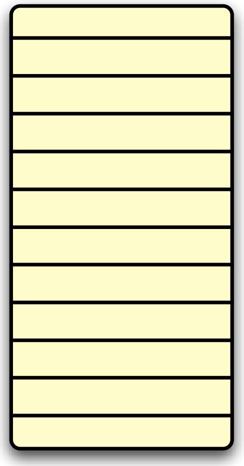
PHY Example: OFDM Tx

WARPLab offline vs. Sysgen real-time

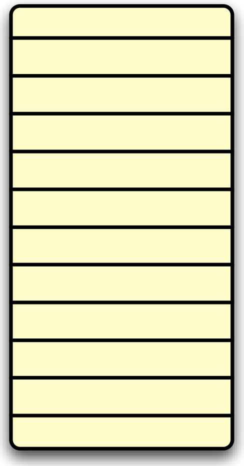
```
trainingSym_freq = [0 -1 1 -1 1 -1 1 -1 1 -1 -1 -1 -1 1 1 -1 1 1 1 1 1 -1 -1 1 1 -1 -1 0 0 0...
 0 0 0 0 0 0 0 0 1 -1 1 1 1 -1 -1 -1 -1 1 1 -1 1 -1 1 1 -1 1 1 1 -1 -1 -1 -1 -1 -1];
trainingSyms_time = ifft(trainingSym_freq, 64);
numBytes = 240;
numOFDMSyms = 20;
fftWindowOffset = 6;
numTrainingSyms = 4;
pilotTonesTx = (randint(numOFDMSyms, 4, 2) .* 2) - 1;

dataTx_qpsk = randint(1, numBytes*4, 4);
mod_qpsk = qammod([0:3], 4, 0, 'gray');
modnorm_qpsk = modnorm(mod_qpsk, 'avpow', 1);
symsTx_qpsk = repmat([zeros(1,47) 1], 1, 20);%qammod(dataTx_qpsk, 4, 0, 'gray') .* modnorm_qpsk;
symsTx_qpsk = qammod(dataTx_qpsk, 4, 0, 'gray') .* modnorm_qpsk;
symsTx_mat = reshape(symsTx_qpsk, 48, numOFDMSyms).'; %20x48
ifftInTx = zeros(20, 64);

%Start filling in the FFT input matrix, leaving zeros where needed
% Columns: 1->DC, 32->max +freq, 33->max -freq, 64->min -freq
% Leave zeros in columns([1 8 22 28:32 33:38 44 58])
ifftInTx(:, [2:7 9:21 23:27 39:43 45:57 59:64]) = symsTx_mat; %20x64
ifftInTx(:, [8 22 44 58]) = pilotTonesTx;
ifftInTx = [repmat(trainingSym_freq, numTrainingSyms, 1); ifftInTx];
ifftOutTx = ifft(ifftInTx.', 64).';

cycExtTx = [ifftOutTx(:,49:64) ifftOutTx];
TxVec = reshape(cycExtTx.', 1, prod(size(cycExtTx)));

TxVec_air = [zeros(1,250) interp(TxVec, 4) zeros(1,2^14 - length(TxVec)*4 - 250)];
```

IFFT

Final Output to DACs



I Buffer



Q Buffer

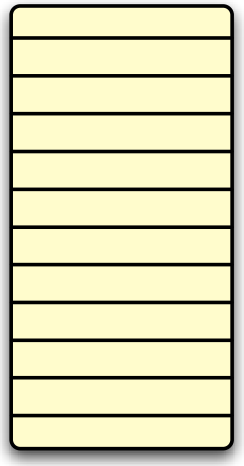
PHY Example: OFDM Tx

WARPLab offline vs. Sysgen real-time

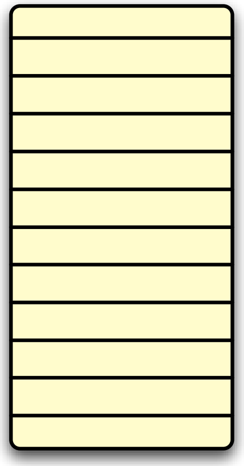
```
trainingSym_freq = [0 -1 1 -1 1 -1 1 -1 1 -1 -1 -1 -1 1 1 -1 1 1 1 1 1 -1 -1 1 1 -1 -1 0 0 0...
 0 0 0 0 0 0 0 0 1 -1 1 1 1 -1 -1 -1 -1 1 1 -1 1 -1 1 1 -1 1 1 1 -1 -1 -1 -1 -1 -1];
trainingSyms_time = ifft(trainingSym_freq, 64);
numBytes = 240;
numOFDMSyms = 20;
fftWindowOffset = 6;
numTrainingSyms = 4;
pilotTonesTx = (randint(numOFDMSyms, 4, 2) .* 2) - 1;

dataTx_qpsk = randint(1, numBytes*4, 4);
mod_qpsk = qammod([0:3], 4, 0, 'gray');
modnorm_qpsk = modnorm(mod_qpsk, 'avpow', 1);
symsTx_qpsk = repmat([zeros(1,47) 1], 1, 20);%qammod(dataTx_qpsk, 4, 0, 'gray') .* modnorm_qpsk;
symsTx_qpsk = qammod(dataTx_qpsk, 4, 0, 'gray') .* modnorm_qpsk;
symsTx_mat = reshape(symsTx_qpsk, 48, numOFDMSyms).'; %20x48
ifftInTx = zeros(20, 64);

%Start filling in the FFT input matrix, leaving zeros where needed
% Columns: 1->DC, 32->max +freq, 33->max -freq, 64->min -freq
% Leave zeros in columns([1 8 22 28:32 33:38 44 58])
ifftInTx(:, [2:7 9:21 23:27 39:43 45:57 59:64]) = symsTx_mat; %20x64
ifftInTx(:, [8 22 44 58]) = pilotTonesTx;
ifftInTx = [repmat(trainingSym_freq, numTrainingSyms, 1); ifftInTx];
ifftOutTx = ifft(ifftInTx.', 64).';

cycExtTx = [ifftOutTx(:,49:64) ifftOutTx];
TxVec = reshape(cycExtTx.', 1, prod(size(cycExtTx)));

TxVec_air = [zeros(1,250) interp(TxVec, 4) zeros(1,2^14 - length(TxVec)*4 - 250)];
```

Cyclic extension

Final Output to DACs



I Buffer



Q Buffer

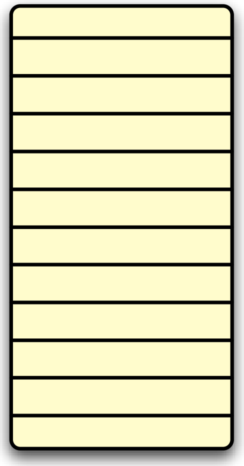
PHY Example: OFDM Tx

WARPLab offline vs. Sysgen real-time

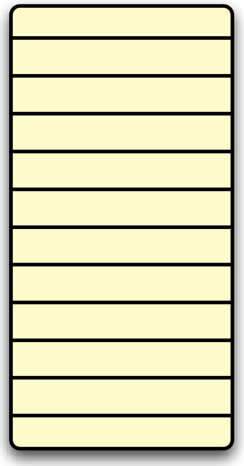
```
trainingSym_freq = [0 -1 1 -1 1 -1 1 -1 1 -1 -1 -1 -1 1 1 -1 1 1 1 1 1 -1 -1 1 1 -1 -1 0 0 0...
 0 0 0 0 0 0 0 0 1 -1 1 1 1 -1 -1 -1 -1 1 1 -1 1 -1 1 1 -1 1 1 1 -1 -1 -1 -1 -1 -1];
trainingSyms_time = ifft(trainingSym_freq, 64);
numBytes = 240;
numOFDMSyms = 20;
fftWindowOffset = 6;
numTrainingSyms = 4;
pilotTonesTx = (randint(numOFDMSyms, 4, 2) .* 2) - 1;

dataTx_qpsk = randint(1, numBytes*4, 4);
mod_qpsk = qammod([0:3], 4, 0, 'gray');
modnorm_qpsk = modnorm(mod_qpsk, 'avpow', 1);
symsTx_qpsk = repmat([zeros(1,47) 1], 1, 20);%qammod(dataTx_qpsk, 4, 0, 'gray') .* modnorm_qpsk;
symsTx_qpsk = qammod(dataTx_qpsk, 4, 0, 'gray') .* modnorm_qpsk;
symsTx_mat = reshape(symsTx_qpsk, 48, numOFDMSyms).'; %20x48
ifftInTx = zeros(20, 64);

%Start filling in the FFT input matrix, leaving zeros where needed
% Columns: 1->DC, 32->max +freq, 33->max -freq, 64->min -freq
% Leave zeros in columns([1 8 22 28:32 33:38 44 58])
ifftInTx(:, [2:7 9:21 23:27 39:43 45:57 59:64]) = symsTx_mat; %20x64
ifftInTx(:, [8 22 44 58]) = pilotTonesTx;
ifftInTx = [repmat(trainingSym_freq, numTrainingSyms, 1); ifftInTx];
ifftOutTx = ifft(ifftInTx.', 64).';

cycExtTx = [ifftOutTx(:,49:64) ifftOutTx];
TxVec = reshape(cycExtTx.', 1, prod(size(cycExtTx)));

TxVec_air = [zeros(1,250) interp(TxVec, 4) zeros(1,2^14 - length(TxVec)*4 - 250)];
```

Filtering

Final Output to DACs



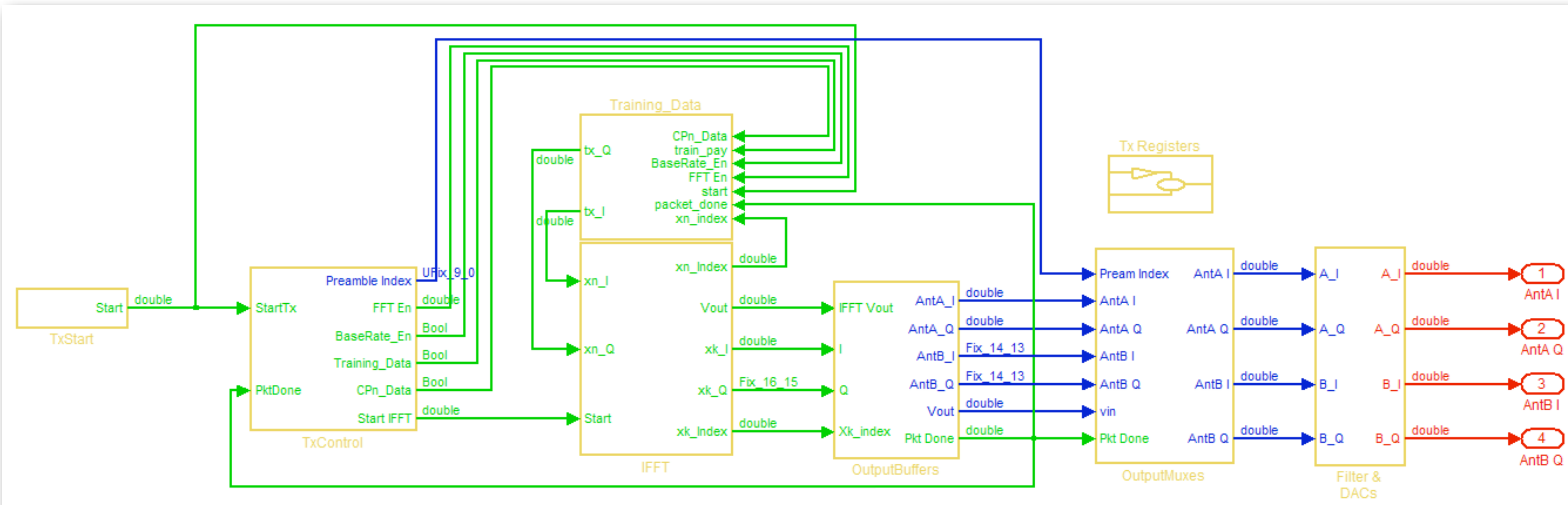
I Buffer



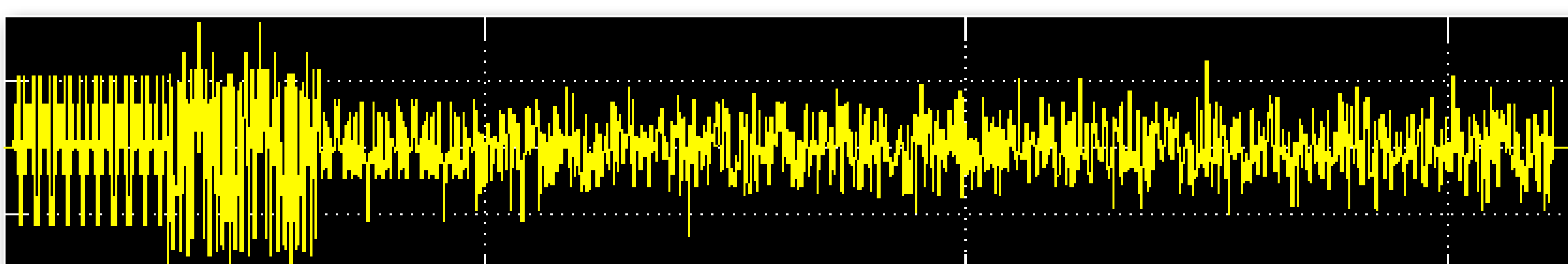
Q Buffer

PHY Example: OFDM Tx

WARPLab offline vs. Sysgen real-time

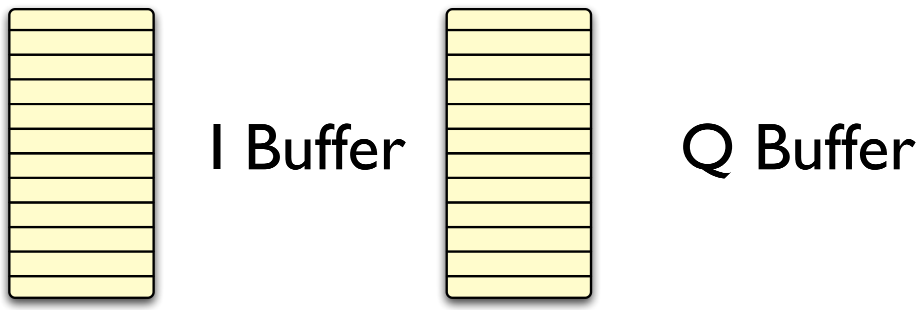


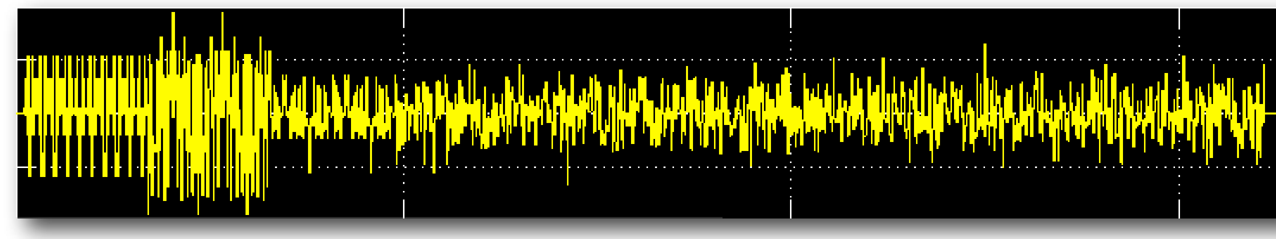
Final Output to DACs



PHY Example: OFDM Tx

WARPLab offline vs. Sysgen real-time





Same final output to DACs

- Signal processing offline MATLAB
- SYNC signal
- Limited buffer length
- PHY algorithm test

- Cost of generating each sample in real time
- Control, Pkt Detection
- Allows a real time system
- Network test

WARPLab Examples

- Hardware characterization
- Channel measurement
- Sphere detection
- Cooperative communications
- Beamforming

Lab 1: WARPLab

- WARPLab graphical interface
- SISO communication
- Measuring the wireless channel
- Building a real bits-to-RF transmitter
- Continuous transmitter mode
- Two-way communication
- 2x2 MIMO communication

WARPLab Overview