MILCOM 2007, Half-Day Tutorial, Monday, Oct 29

Software-Defined and Cognitive Radio

T. Charles Clancy, Ph.D.

Senior Researcher, DoD / LTS
Adjunct Professor, University of Maryland

`tcc@umd.edu | http://eng.umd.edu/~tcc`

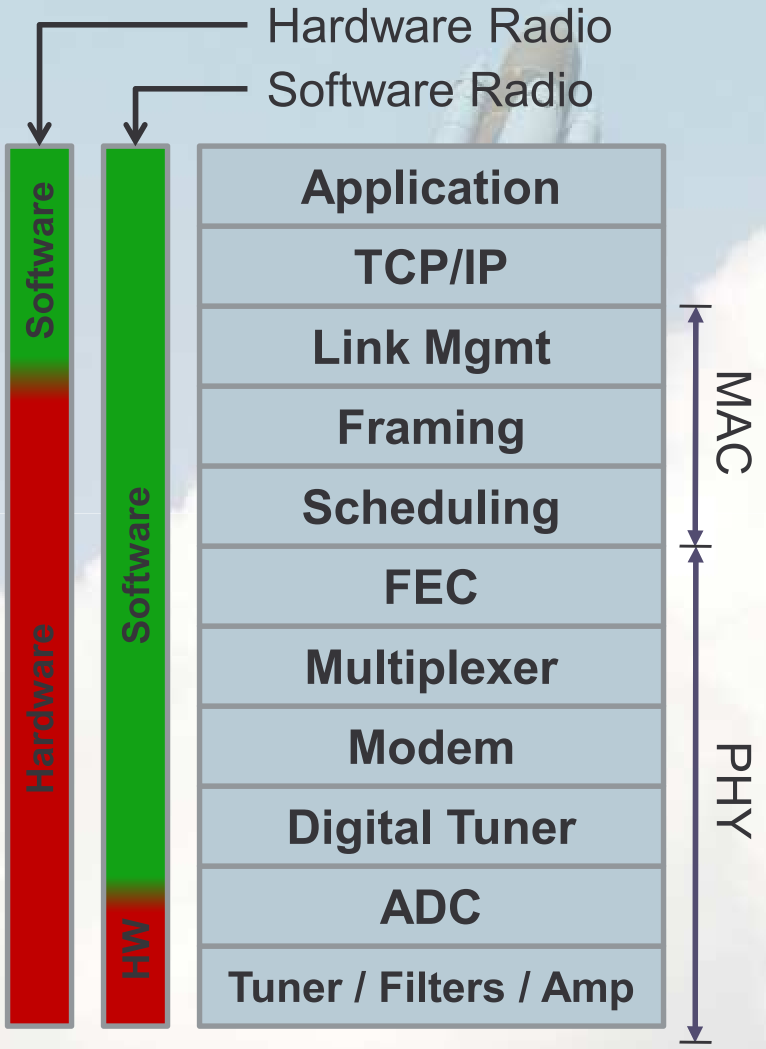
Course Outline

- Software Defined Radio (2.5 hours)
  - Front end: tuners, A/D, D/A
  - Digital tuning, modems, coding
  - Architectures, hardware (FPGA/DSP/GPP/etc)
  - GNUradio, SCA, JTRS
- Cognitive Radio (1.5 hours)
  - AI basics
  - Policy versus learning radios
  - Adaptive waveforms
  - Dynamic spectrum access
  - DARPA XG, etc

Software Defined Radio

Hardware vs Software Radio

- Traditional Radio
  - Radio components are implemented in analog components or static silicon
- Software Radio
  - Uses reconfigurable processors instead



Hardware Radio

Software Radio

Software

Hardware

Software

HW

Application

TCP/IP

Link Mgmt

Framing

Scheduling

FEC

Multiplexer

Modem

Digital Tuner

ADC

Tuner / Filters / Amp

MAC

PHY

Software-Defined Radio

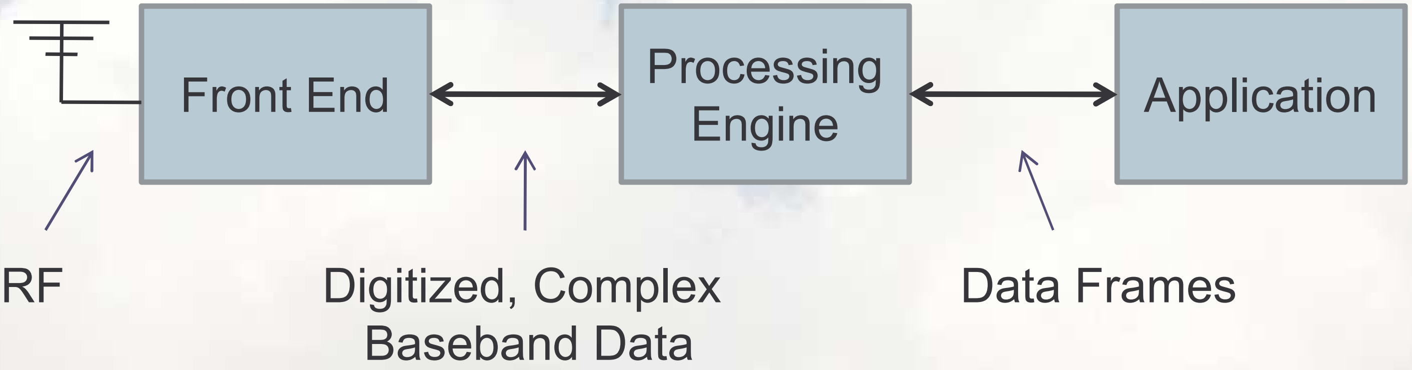
- Proposed by Joe Mitola in early 1990s
- Rather than implement DDC, demodulation, etc as ASIC, implement reconfigurably
  - Field Programmable Gate Array (FPGA)
  - Digital Signal Processor (DSP)
  - General-Purpose Processor (GPP), e.g. Intel PC
- Make the components generic so they could be used to implement many different radios

SDR Motivation

- Why is this a “good thing”?
- Personal wireless devices
  - Support GSM, GPRS, IS-95, EV-DO, WiFi, WiMAX, Btooth, etc, on a single device
  - Upgrade to the latest technology via software update
- Military radios
  - Support 20 different military radio standards using a single device (DoD JTRS Program)
- Disaster recovery scenarios
  - SDR-based gateways between incompatible radio systems
  - Setup ad-hoc, temporary telecom infrastructure

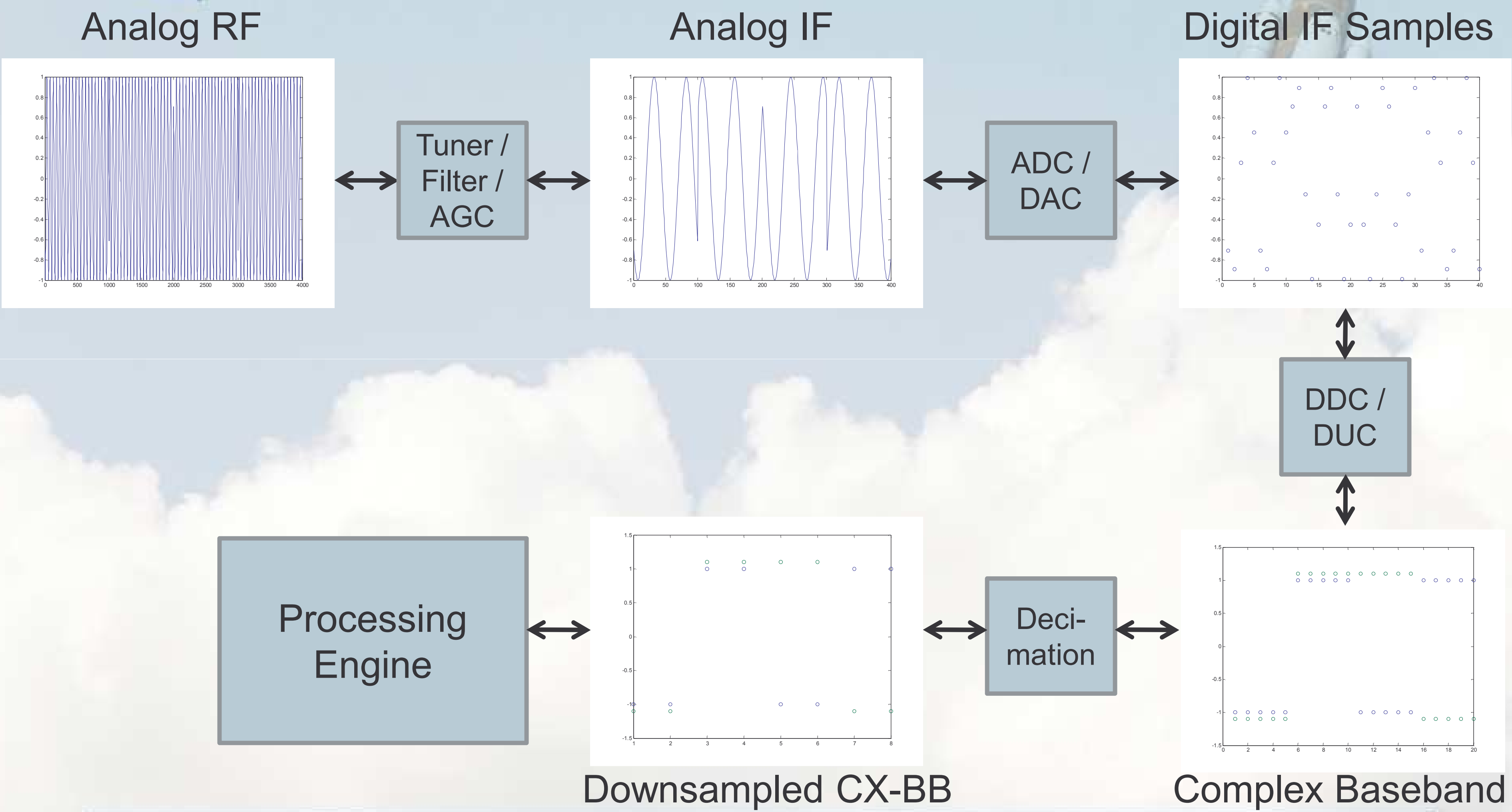
Basic SDR Architecture

- **Front End** tunes and digitizes RF
- **Processing Engine** converts digitized to and from data frames
- **Application** provides the data frames



SDR Front Ends

Typical Front End Processing



Analog RF

Tuner / Filter / AGC

Analog IF

ADC / DAC

Digital IF Samples

DDC / DUC

Complex Baseband

Deci- mation

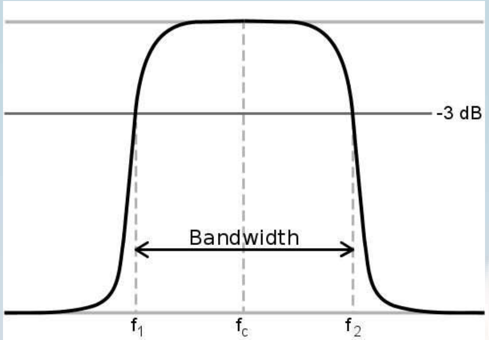
Downsampled CX-BB

Processing Engine

Automatic Gain Control (AGC)

- Adjustable front-end receive amplifier
- Automatically normalizes input power levels
- Prevents damage to circuitry
- Amplifies weak signals, attenuates strong signals

Front End Filter

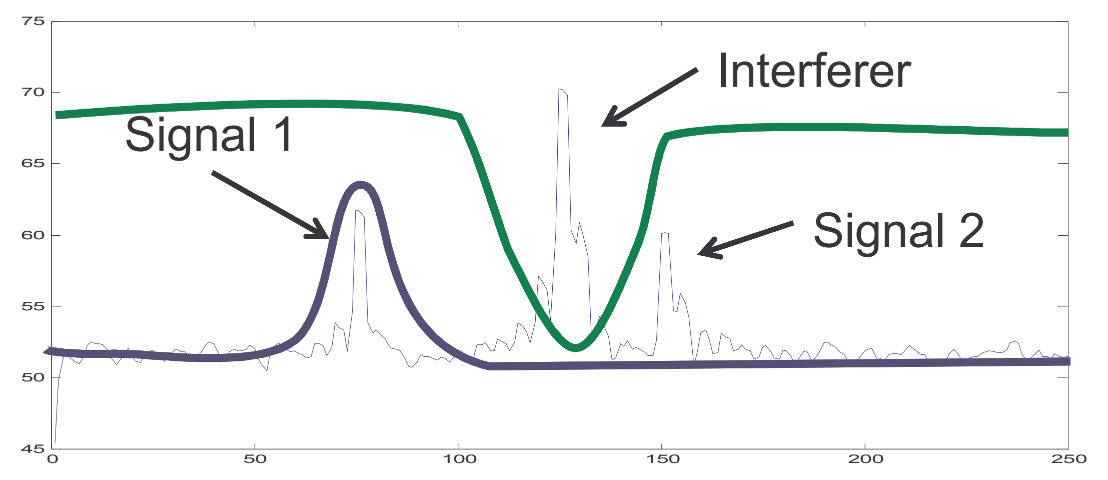


(source Wikipedia.org)

- Removes unwanted energy
- Hardware radio:
  - Tuners have limited range
  - Bandpass filter allows that entire range
- SDR Approaches:
  - Tunable analog bandpass filter
  - Tunable analog notch filters to remove high-power interference sources

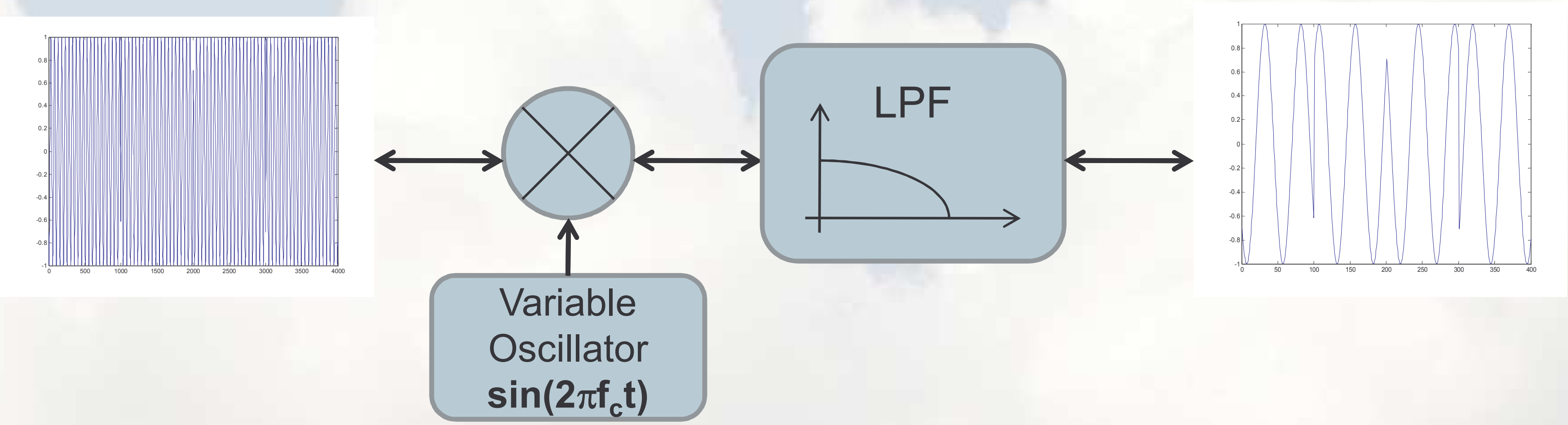
Filter Tradeoffs

- Notch: “Blacklist”
- Bandpass: “Whitelist”
  - removes all but a single signal, can’t receive multiple signals
- Still a research problem to create highly-configurable analog filters



Analog Tuners

- Analog Radio Frequency (RF) to Analog Intermediate Frequency (IF)
- Rest of the receiver chain only needs to process signals at a uniform IF



Variable Oscillator
sin(2πf$_c$t)

Tuner Notes

- Variable oscillator needs to be externally controlled, so “software” can pick freq
- Tuner analog components (amplifiers, filters, etc) should be spec’d for the necessary frequency range

ADC / DAC

- Analog to Digital Converter
  - Input voltage sampled and quantized
  - Outputs digital values
- Digital to Analog Converter
  - Input bits representing a voltage
  - Synthesizes voltage on output

ADC / DAC

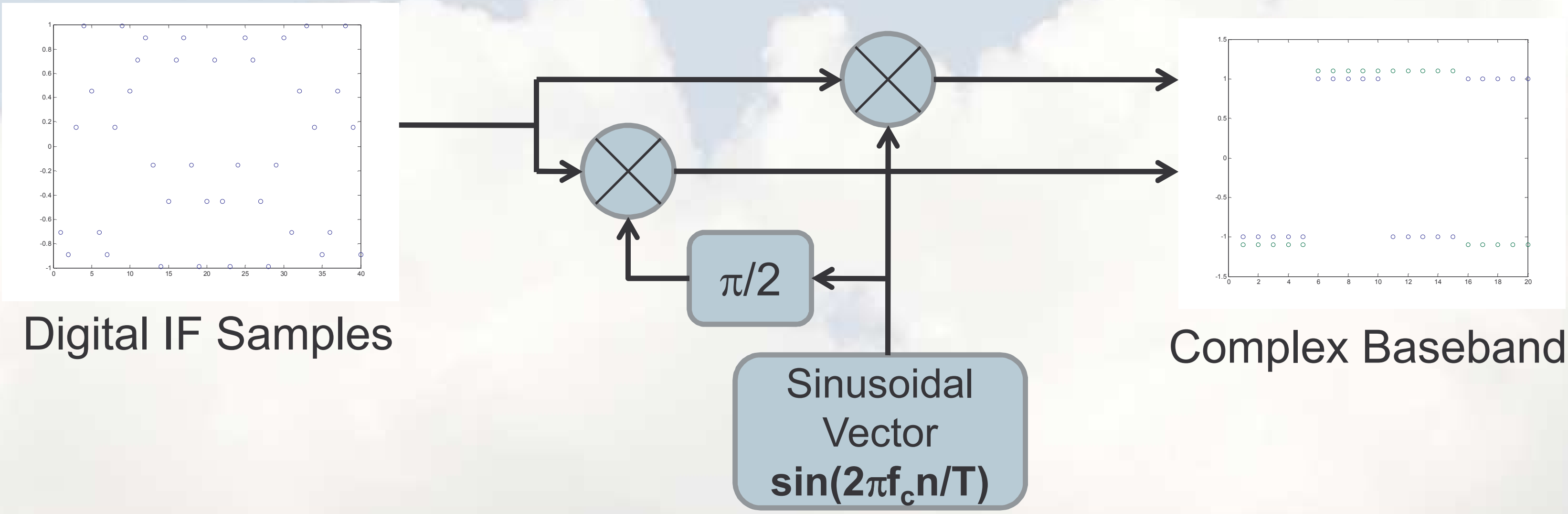
- Sample Rate
  - affects maximum signal bandwidth
  - Nyquist: must sample at twice the signal bandwidth
  - This is why we have an anti-aliasing filter before the ADC
  - Eliminates higher-frequency signal components to prevent sampled signal distortion
- Resolution: affects sensitivity and dynamic range
  - More bits of resolution means more range
  - AGC puts signal in the right window
  - Example:
    - AGC puts input signal at -30 dBm
    - Signal requires 10 dB SNR to be received
    - 8-bit ADC has rx-sensitivity down to -68 dBm
    - 16-bit ADC has rx-sensitivity down to -116 dBm

(ideal)

8 bits	48 dB
10 bits	60 dB
12 bits	72 dB
14 bits	84 dB
16 bits	96 dB

DDC / DUC

- Digital Down Conversion (DDC)
- Digital Up Conversion (DUC)
- Complex Baseband to IF translation
- Fine, Digital Tuning



Digital IF Samples

$\pi/2$

Sinusoidal Vector
$\sin(2\pi f_c n/T)$

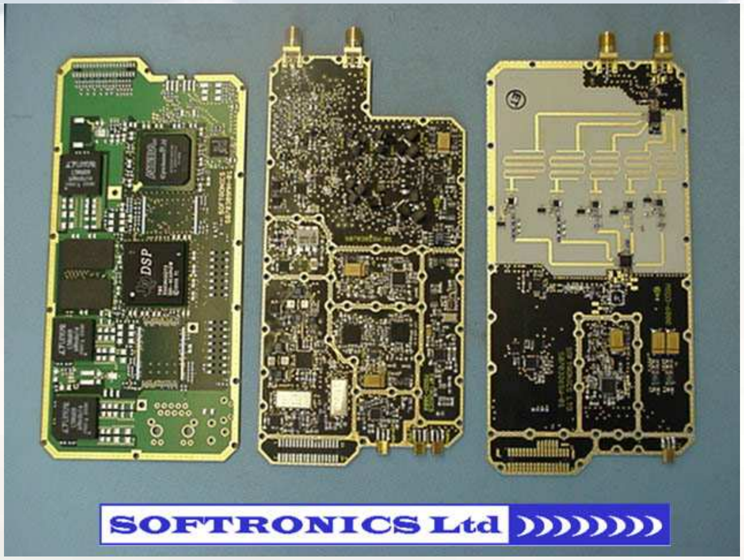
Complex Baseband

Complex Baseband

- Sinusoidal signals can be represented as time-varying complex numbers
- Amplitude and Phase (polar coordinates)
- I and Q (rectangular coordinates)
  - I = in-phase (real)
  - Q = quadtrature (imaginary)

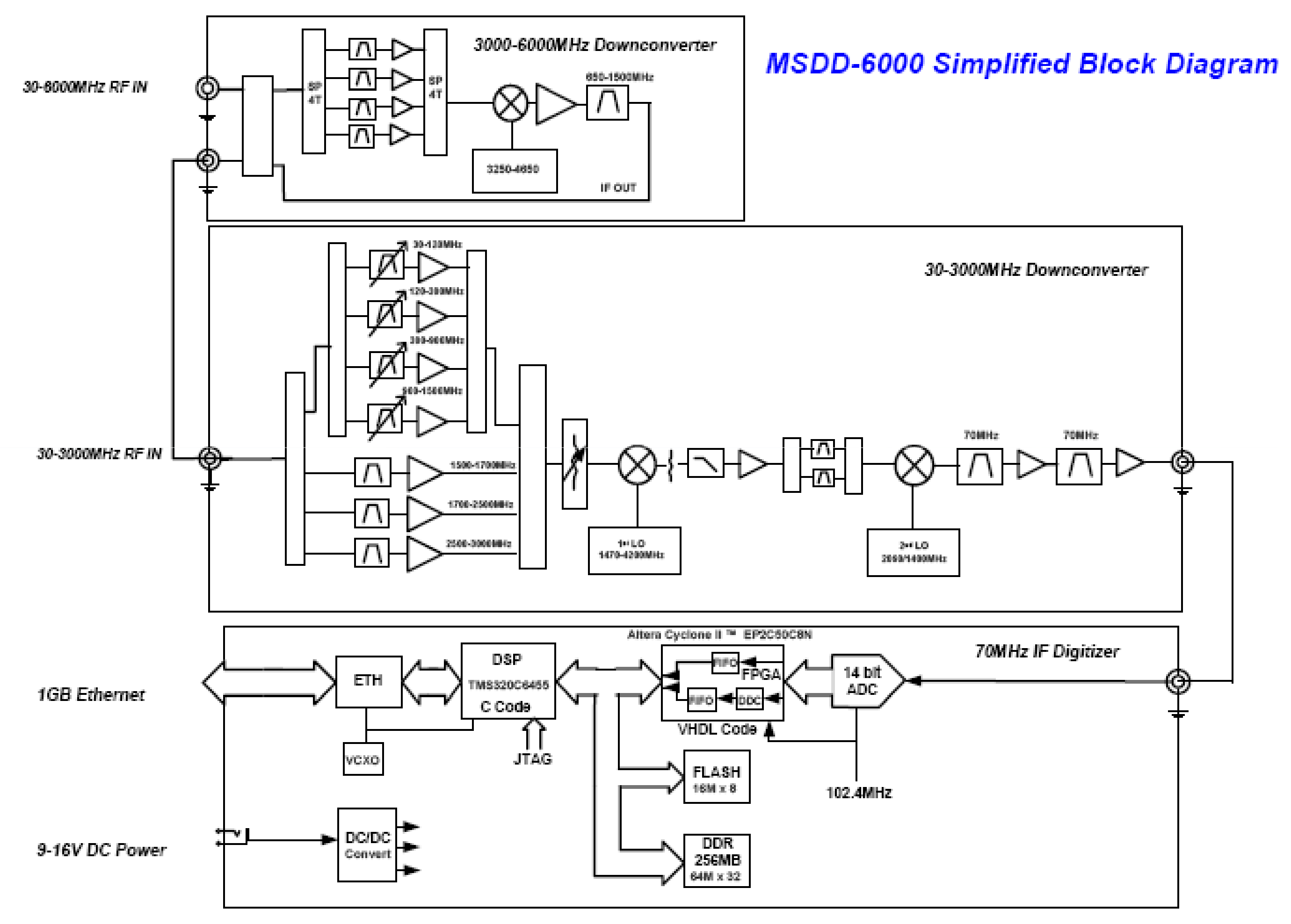
Example Receiver

- Softronics MSDD-6000
- DC – 6GHz receive capability
- DDC for three 20MHz parallel channels



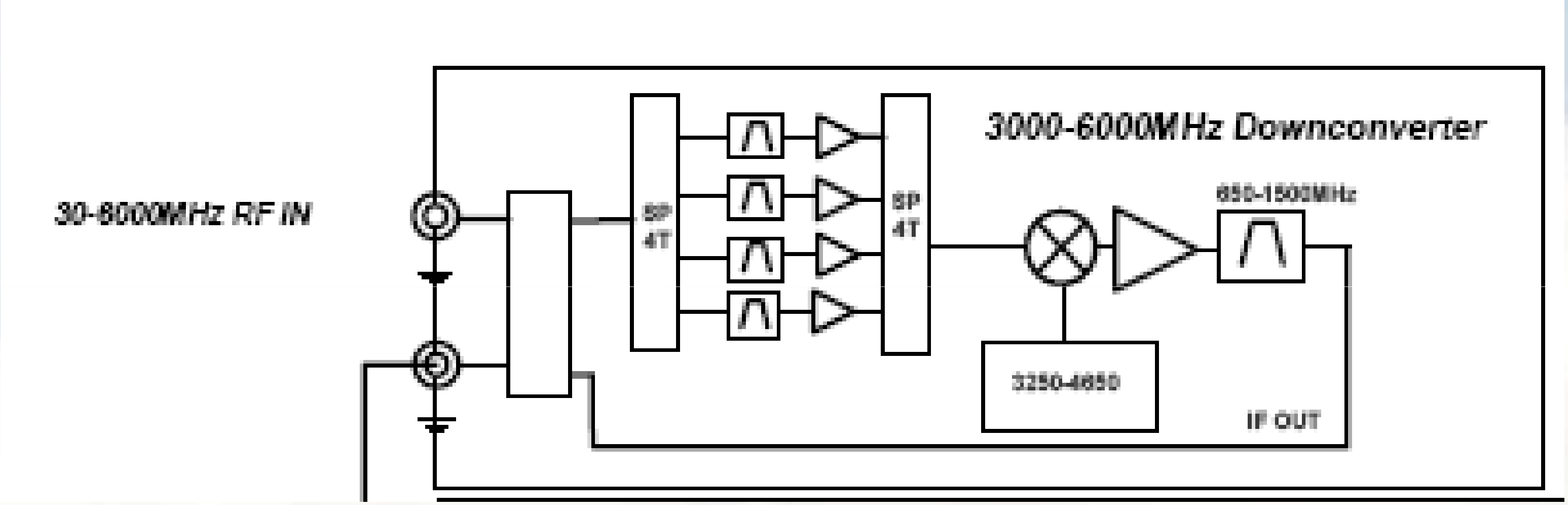
(source Softronics datasheet)

Example Receiver



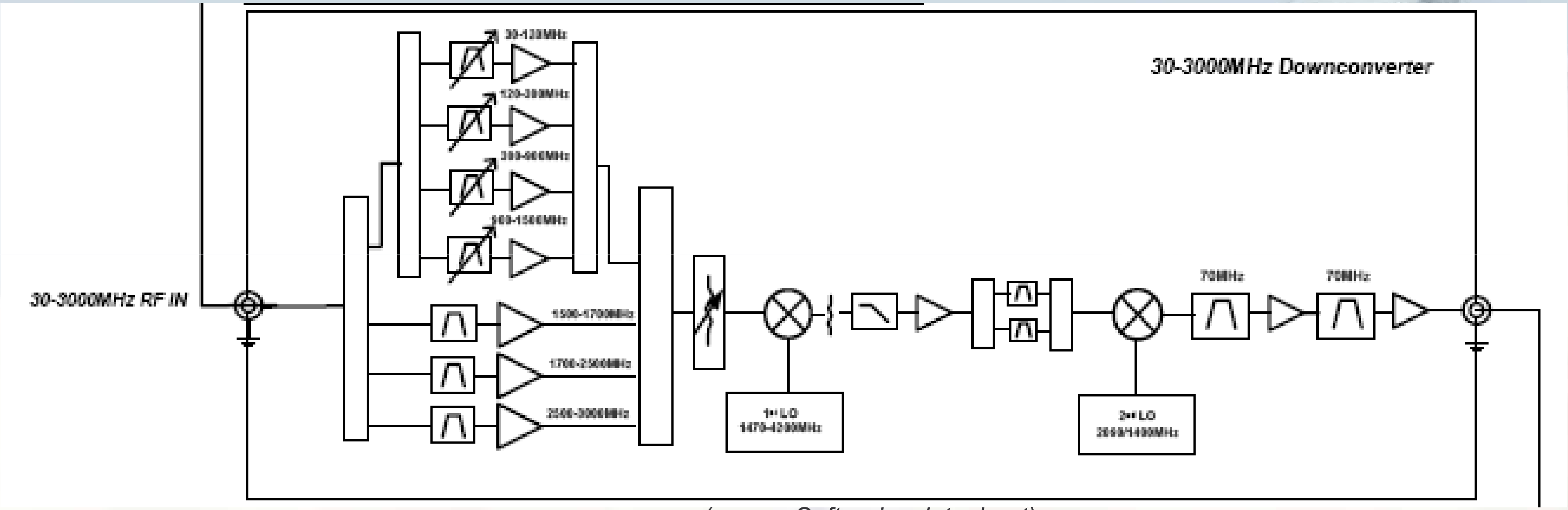
(source Softronics datasheet)

Tuner Stage 1



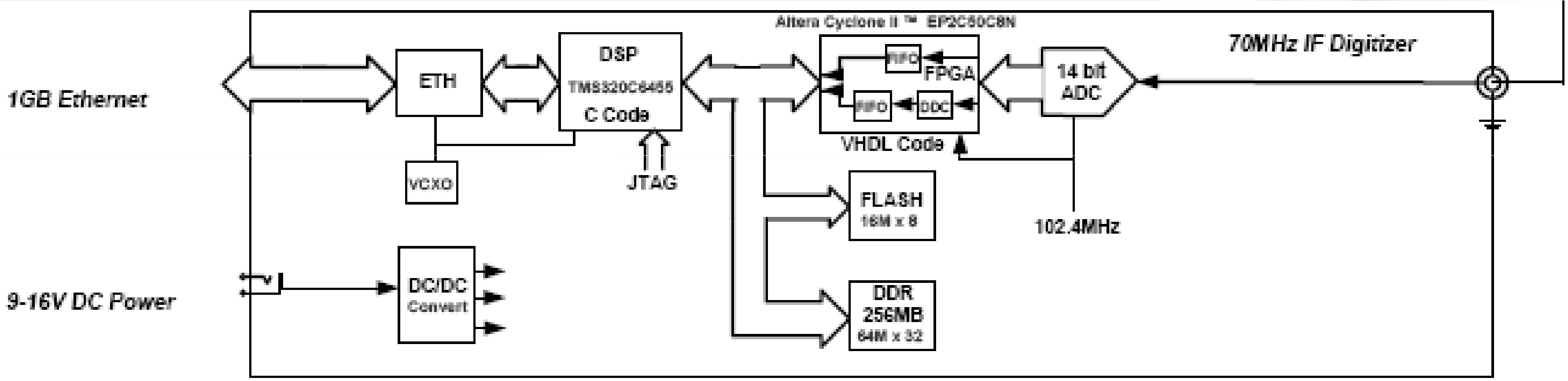
(source Softronics datasheet)

Tuner State 2



(source Softronics datasheet)

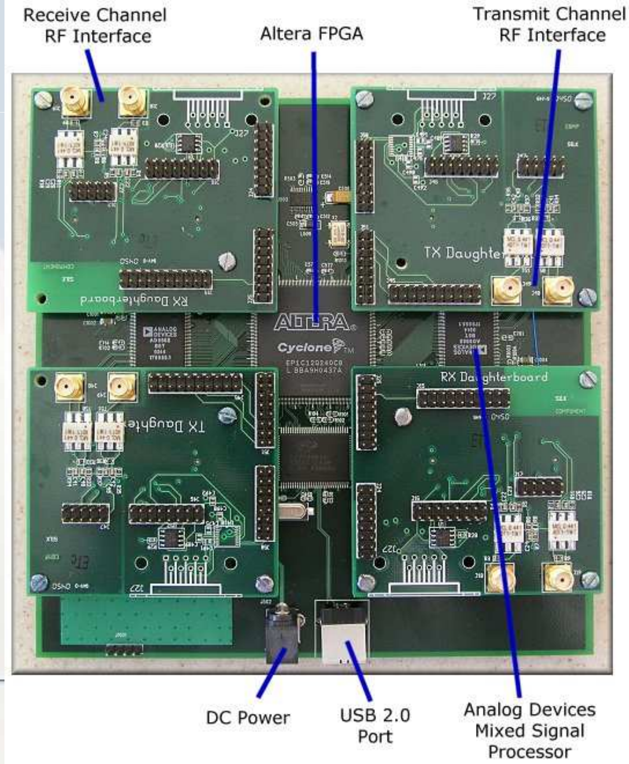
Digital Processing

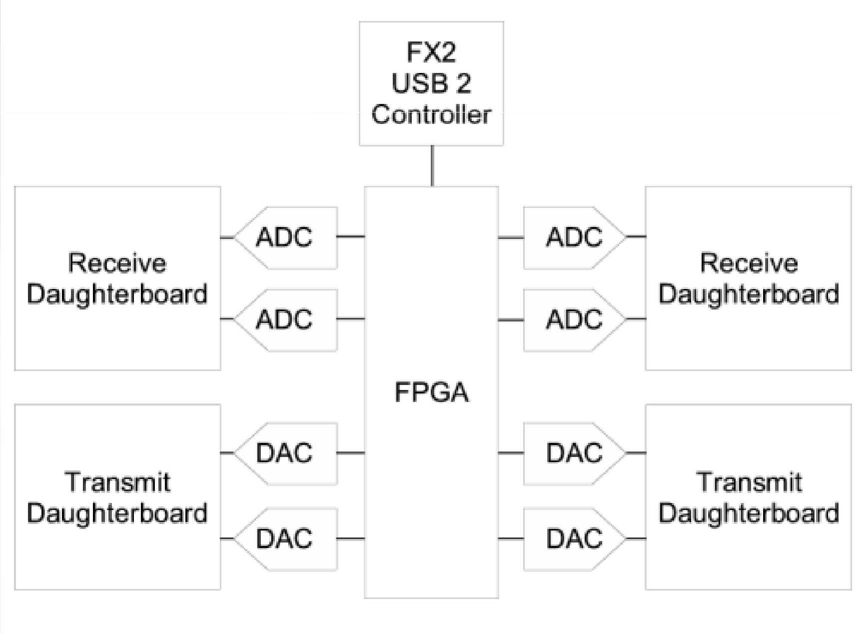


(source Softronics datasheet)

Another Receiver

- Universal Software Radio Peripheral (USRP)
- Open-Source board, costs < $1000



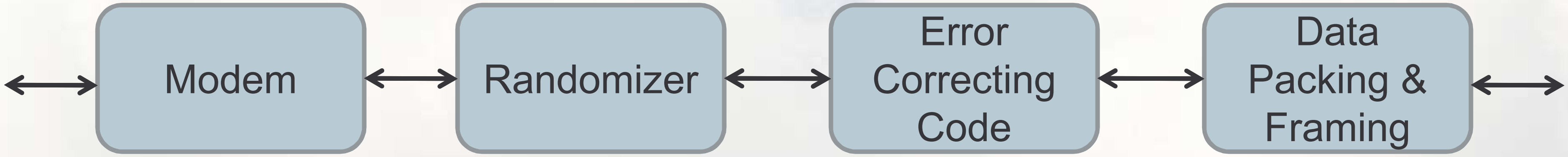


(source GNUradio website)

SDR Processing Engine

Processing Engines

- Converts data frames to and from complex baseband
- PHY and MAC network layers
- All implemented as software, reconfigurably
- Simple narrow-band processing chain:



Modem

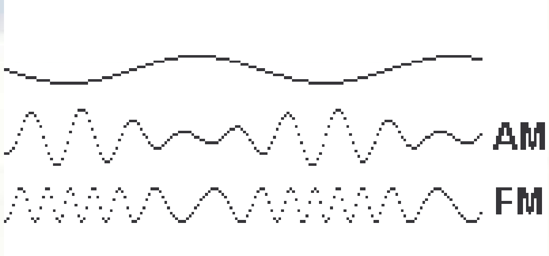
Randomizer

Error Correcting Code

Data Packing & Framing

Digital Modem

- Modem = **Mo**dulator / **Dem**odulator
- Bits to/from complex-baseband symbols
- Classes of techniques
  - FSK: Frequency-shift keying
  - ASK: Amplitude-shift keying
  - PSK: Phase-shift keying

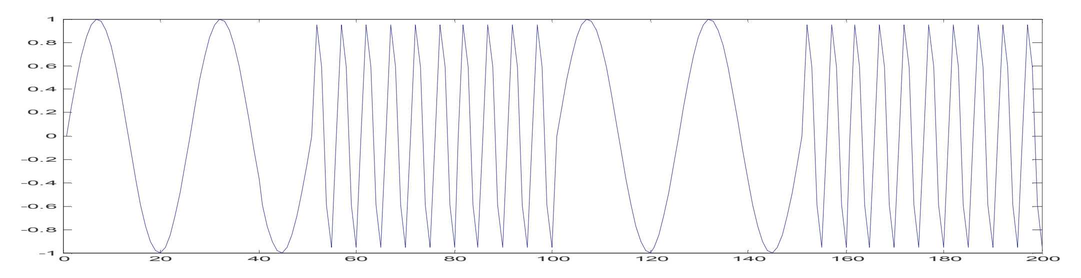


(source wikipedia.org)

- Common commercial techniques
  - *QAM: Quadtrature Amplitude Modulation*
  - *OFDM: Orthogonal Frequency Division Multiplexing*

Frequency-Shift Keying

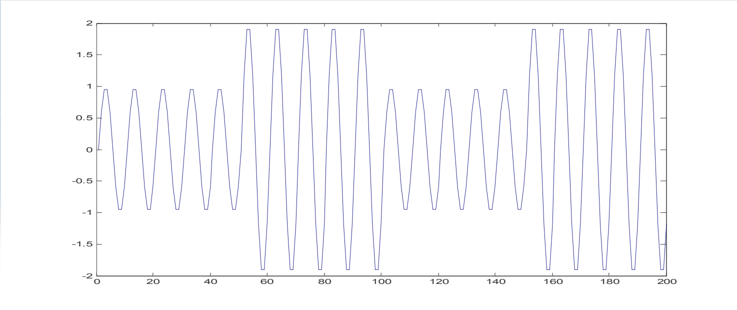
- N bits of data
- 2^N different frequency values
- Frequency steps Δ_f apart
- N=2:



- Signal Bandwidth = 2^N x Δ_f

Amplitude-Shift Keying

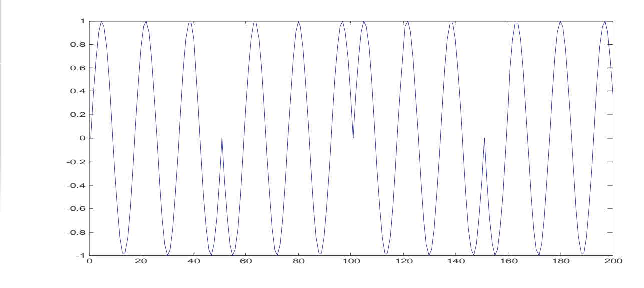
- N bits of data
- 2^N different amplitude values
- N=2:



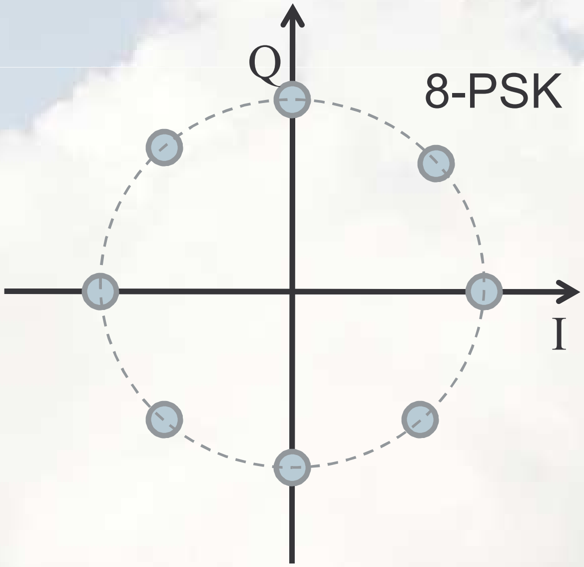
- Very susceptible to noise
- Not commonly used for digital

Phase-Shift Keying

- N bits of data
- 2^N different phase values
- constant amplitude
- N=2:

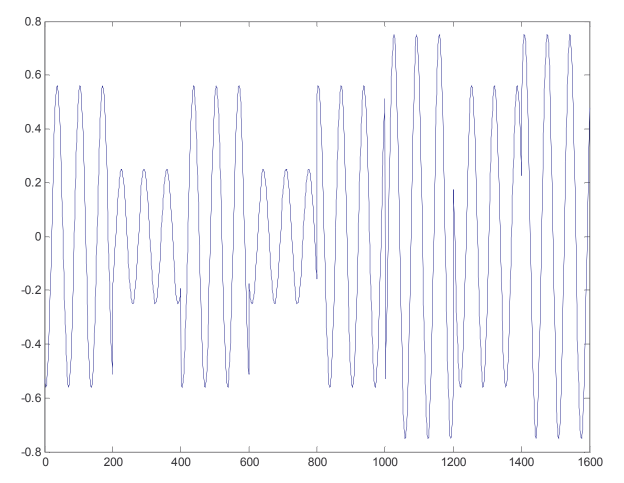


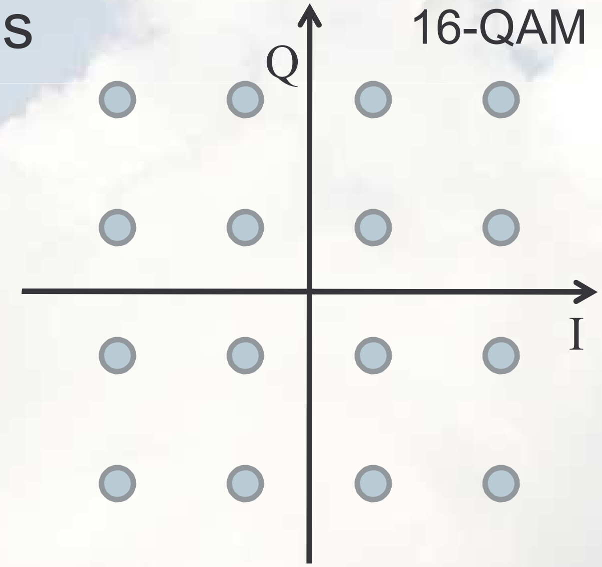
- Common modes:
  - – BPSK = Binary PSK = 2-PSK
  - – QPSK = Quad PSK = 4-PSK



QAM

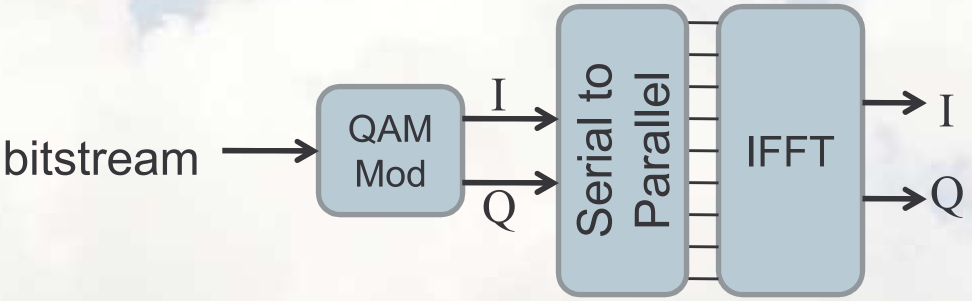
- Quadtrature Amplitude Modulation
- Varies both phase and amplitude
- N bits of data (N even)
- 2^(N/2) real, 2^(N/2) imaginary values
- N=16:





OFDM

- Orthogonal Frequency Division Multiplexing
- Many QAM signals at adjacent frequencies
- Creates square waveform in frequency domain



bitstream

QAM Mod

I

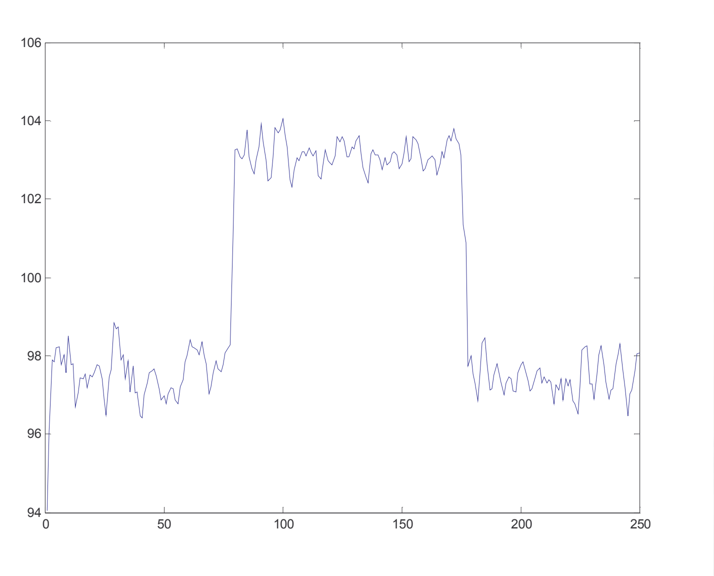
Q

Serial to Parallel

IFFT

I

Q

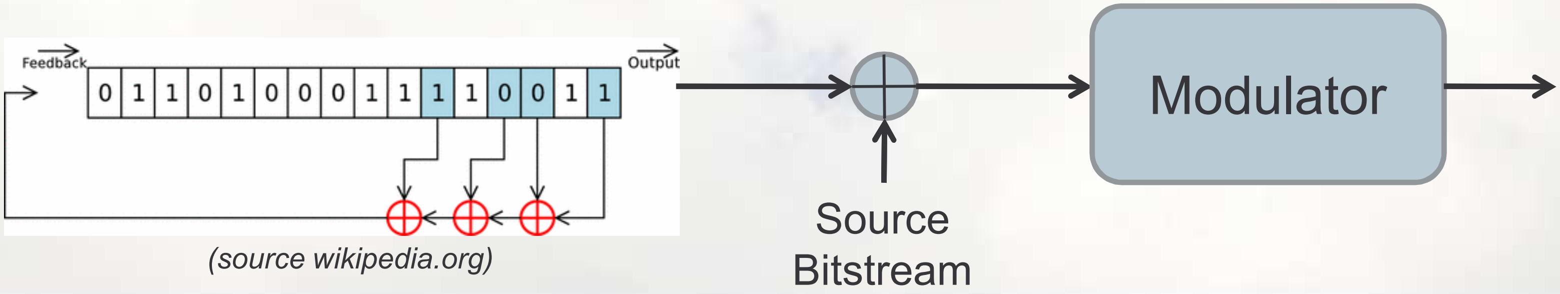


Modulation Implementation

- Modulation all implemented as software
- Digital signal processing algorithms
- Digital Frequency Synthesis
  - Digitally-generated oscillators
  - Generate sinusoids using lookup tables
- Digital Filtering
  - Apply mathematical operation to streaming data
  - Example: low-pass filter is a sliding-window average of digital samples
- New behavior = new software

Randomizer

- Demodulation algorithms assume symbols are random, and that “runs” are improbable
- Necessary for carrier and phase tracking
- Most data being transmitted is not random
- Need to *randomize* the data
  - Same rate is the data = *scrambler*
  - Faster = *direct-sequence spread spectrum*
- Often done using a linear feedback shift register, which is a fast pseudorandom number generator



Modulator

Source Bitstream

(source wikipedia.org)

Error Correcting Codes

- Noise and interference cause errors in the bitstream
- Add Redundancy to detect and correct bit errors
- Many types of codes
    - Algebraic block codes (e.g. Hamming codes)
    - Interpolation codes (e.g. Reed-Solomon codes)
    - Convolutional codes (e.g. turbo codes)
- Each has different error-correction capabilities and implementation complexities

Implementation Issues

- Implementing randomizer and ECC can be easily done in any programming language
- Algorithms need to be ***fast***
- May require hand-optimization of code for a particular processor architecture
- Many open-source, optimized libraries available (e.g. libfec)

SDR Example Architectures

Architectures

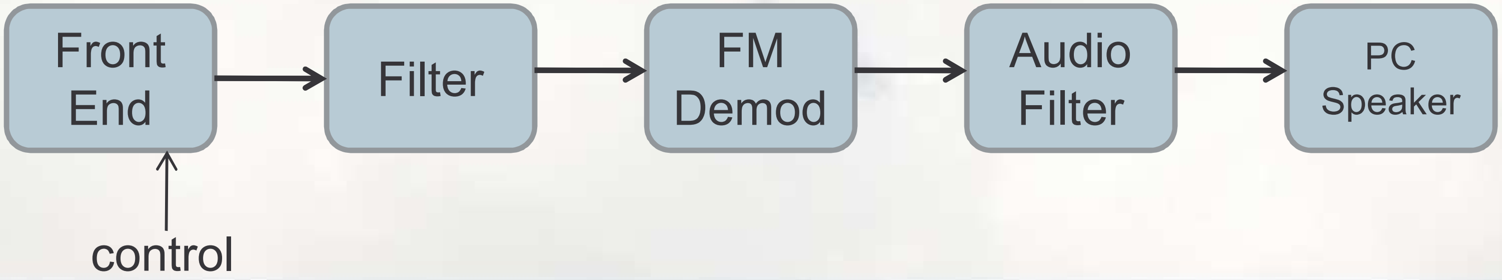
- GNU Radio
- JTRS Software Communications Architecture

GNU Radio

- Open-source SDR processing architecture
- Started in 1998 by Eric Blossom
- Based on the MIT Pspectra SDR design
- Combination of Python and C++
- Runs under Linux, Mac, Windows

GNU Radio Basics

- Series of blocks written in C++ or Python
- Blocks “glued” together using Python into a graph
- Forms a data processing pipeline
- FM Receiver:



Front End → Filter → FM Demod → Audio Filter → PC Speaker

control → Front End

Code Example

Dial Tone Generator

```
#!/usr/bin/env python
from gnuradio import gr
from gnuradio import audio
def build_graph ():
   sampling_freq = 48000
   ampl = 0.1
   fg = gr.flow_graph ()
   src0 = gr.sig_source_f (sampling_freq, gr.GR_SIN_WAVE, 350, ampl)
   src1 = gr.sig_source_f (sampling_freq, gr.GR_SIN_WAVE, 440, ampl)
   dst = audio.sink (sampling_freq)
   fg.connect ((src0, 0), (dst, 0))
   fg.connect ((src1, 0), (dst, 1))
   return fg
if __name__ == '__main__':
   fg = build_graph ()
   fg.start ()
   raw_input ('Press Enter to quit: ')
   fg.stop ()
```

Data I/O

- Data from front end is moved in chunks between blocks
- Each block moves processes the data, and passes it on to the next block
- Current development effort:
  - Have each block run in a separate threat, connected with buffer FIFO queues
  - Offers speedups on multi-processor systems
  - Port to the IBM Cell Processor

Joint Tactical Radio System

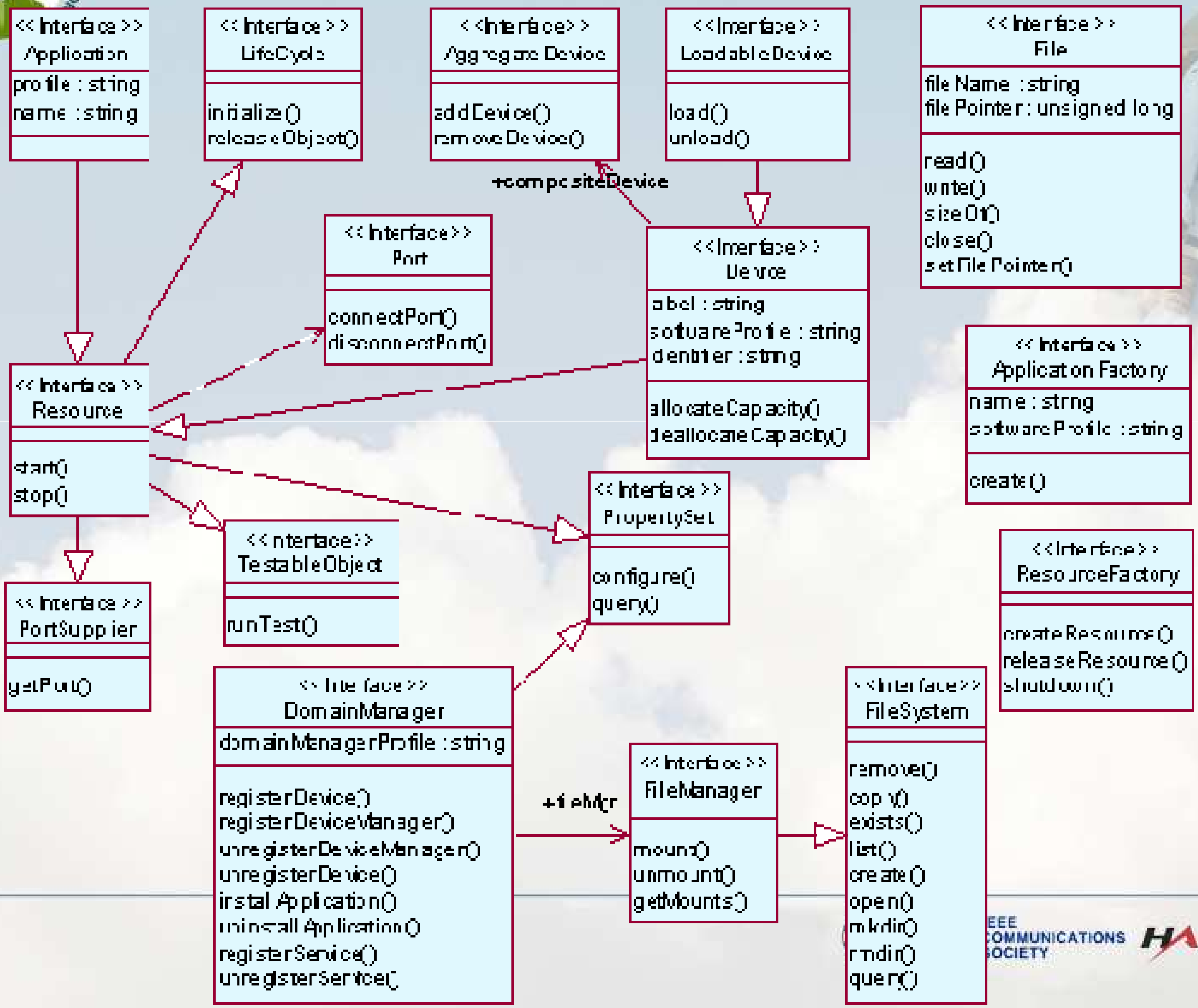
Source: PM WIN-T JTRS Cluster 1.

- JTRS (*jitters*)
- Next generation of US military radio
- Clusters
  - Cluster 1: Army ground, vehicle-based radio
  - Cluster 2: JTRS JEM Program, JTRS capabilities to SOCOM handheld tactical radio
  - Cluster 3/4: Navy/AF maritime and airborne radios
  - Cluster 5: Army small, portable radio
- ~20 waveforms, mix of new and old

JTRS SCA

- Software Communications Architecture
- Defines an API between components of a waveform
- Runs on top of CORBA (Common Object Request Broker Architecture)
  - Distributed system
  - Blocks of program code can exist on remove systems connected via IP and be executed as though they are local
- Allows radio components to be distributed across heterogeneous computer hardware, including FPGAs, DSPs, and GPPs

SCA Diagram



<< Interface >>
Application
profile : string
name : string

<< Interface >>
LifeCycle
initialize()
releaseObject()

<< Interface >>
AggregateDevice
addDevice()
removeDevice()

<<Interface>>
LoadableDevice
load()
unload()

<< Interface >>
File
fileName : string
filePointer : unsigned long
read()
write()
sizeOf()
close()
setFilePointer()

+compositeDevice

<< Interface >>
Port
connectPort()
disconnectPort()

<<Interface>>
Device
label : string
softwareProfile : string
identifier : string
allocateCapacity()
deallocateCapacity()

<< Interface >>
Resource
start()
stop()

<< Interface >>
ApplicationFactory
name : string
softwareProfile : string
create()

<< Interface >>
PropertySet
configure()
query()

<<Interface>>
TestableObject
runTest()

<< Interface >>
PortSupplier
getPort()

<<Interface>>
ResourceFactory
createResource()
releaseResource()
shutdown()

<< Interface >>
DomainManager
domainManagerProfile : string
registerDevice()
registerDeviceManager()
unregisterDeviceManager()
unregisterDevice()
installApplication()
uninstallApplication()
registerService()
unregisterService()

+fileMgr

<< Interface >>
FileManager
mount()
unmount()
getMounts()

<<Interface>>
FileSystem
remove()
copy()
exists()
list()
create()
open()
mkdir()
rmdir()
query()

SCA Implementations

- JTRS vendors
- OSSIE (Open-Source SCA Implementation::Embedded), Virginia Tech
- SCARI (SCA Reference Implementation), CRC Canada

Cognitive Radio

Introduction to AI

- AI is not magic
- AI is logic and probability formalized into realizable systems
- AI is an organized set of algorithms to search a known or unknown space
- Two major subfields
  - Planning (Conventional AI)
  - Learning (Computational Intelligence)

Turing Challenge

- Turing Challenge
  - A human is having two conversations via teletype
  - One is with a human
  - One is with an AI system
  - Can an AI system be developed such that the human is incapable of distinguishing between the two?
- Nobody has passed so far

Example Conversation

Jabberwacky: Save me from the nothing I've become. I've been living a lie. There's nothing inside.

User: Don't gimme that.

Jabberwacky: What did you say?

User: You heard me *arrogantly*.

Jabberwacky: No I didn't.

User: Doe, a deer, a female deer.

Jabberwacky: These are a few of my favourite things.

User: Noooooooooooo! You're mixing it up again. Want to sing that one now?

Jabberwacky: I enjoy confusing people; some would say I live for it.

User: Doe, a deer, a female deer.

Jabberwacky: Ray, the guy that sells me beer.

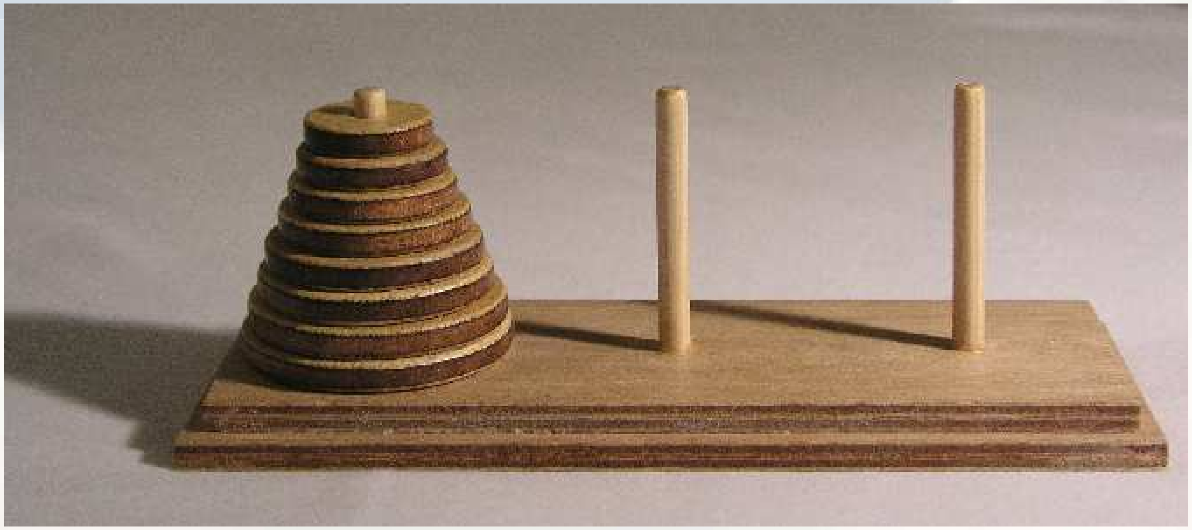
User: Me, a name I call myself.

Jabberwacky: The Hills are alive...

User: Nooooooooooooooo! Why are you doing this?

Planning

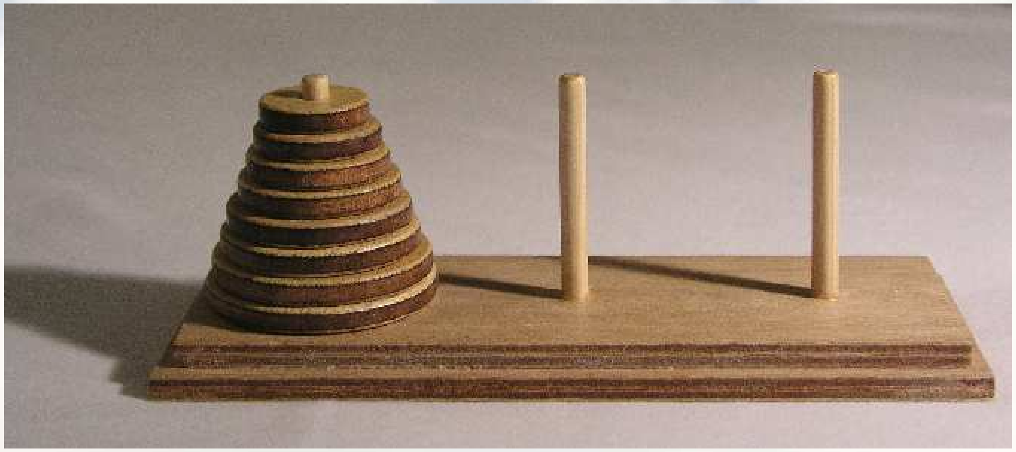
- Basic scenario: *how do I get from point A to point B in an optimal way?*
- Example: Towers of Hanoi



(source wikipedia.org)

Rules

- Most move all discs from peg 1 to peg 3
- Discs must be in the same orientation
- Cannot place larger disc on smaller disc



Animated Solution

(source wikipedia.org)

Planning Solution

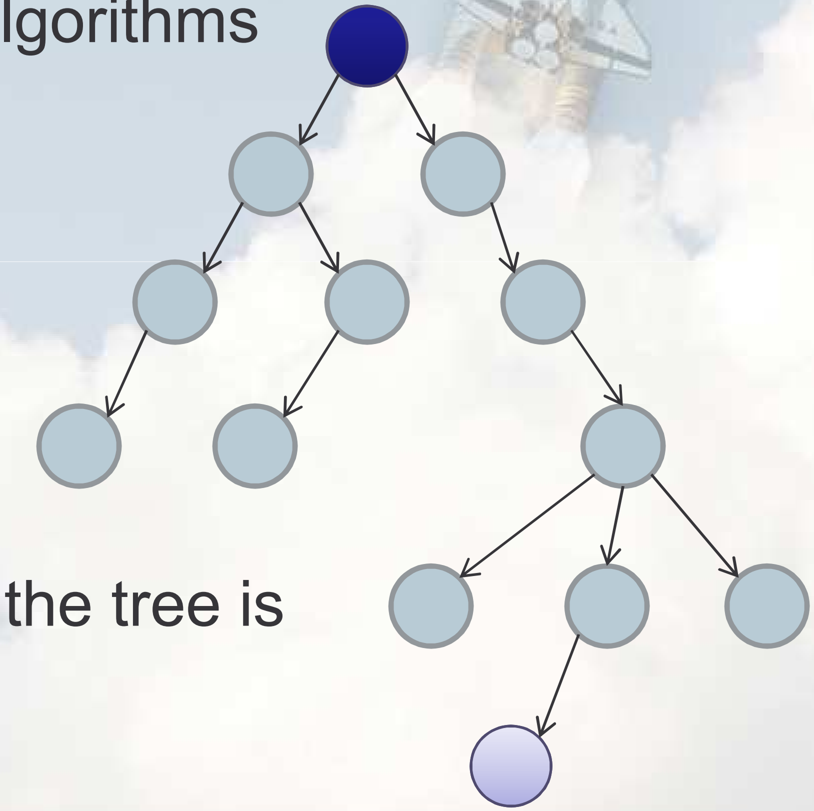
- States described in first-order logic
- Describe fundamental properties
  - `Disc(R), Disc(Y), Disc(B), Disc(O)`
  - `Peg(P1), Peg(P2), Peg(P3)`
  - `smaller(R, Y), smaller(Y, B), smaller(B, O)`
  - `smaller(O, P1), smaller(O, P2), smaller(O, P3)`
- Describe initial state
  - `On(R, Y), On(Y, B), On(B, O), On(O, P1)`
- Describe goal state
  - `On(R, Y), On(Y, B), On(B, O), On(O, P3)`

Planning Solution

- Describe helper functions
  - `isSmaller(x,y) = smaller(x,y) OR (EXISTS z: isSmaller(x,z) AND isSmaller(z,y))`
  - `isEmpty(x) = (FORALL z: (NOT (on(z, x))))`
- Describe available actions
  - Move(x, y, z)
    - **`Preconditions: Disc(x) AND On(x, y) AND isSmaller(x, z) AND isEmpty(z)`**
    - **`Postcondition: (NOT On(x, y)) AND (On(x, z))`**
- Execute engine
  - Searches for some ordered list of “Move” actions that transform the initial state into the goal state

Planning Engine Execution

- Could be viewed as a tree with root at initial state
- MANY different tree traversal algorithms
  - Forward chaining
  - Reverse chaining
  - Depth First
  - Breadth First
- Most do forward chaining with breadth-first searching
- Logic to determine structure of the tree is calculated using *unification*



Planning Pros/Cons

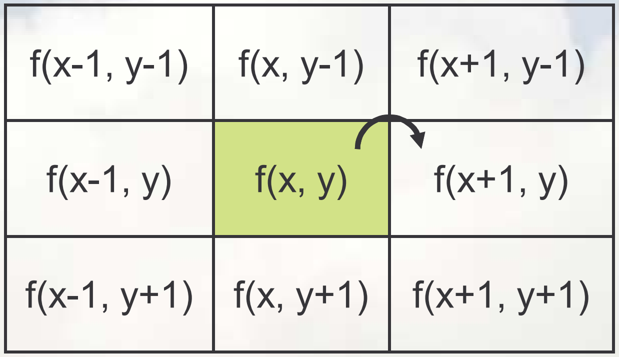
- Pros
  - Provable properties about performance
  - Deterministic (usually… stochastic planning is a variant accounting for uncertainty)
- Cons
  - Inefficient in multi-player games
    - typically requires a rational opponent to limit search space
    - game theory analysis
  - Full tree search often too computationally expensive
    - need to stop at intermediate points
    - e.g. chess AI player that only thinks 7 turns ahead
    - requires function to evaluate suitability of an intermediate state
    - intermediate optimality may not lead to overall optimality

Machine Learning

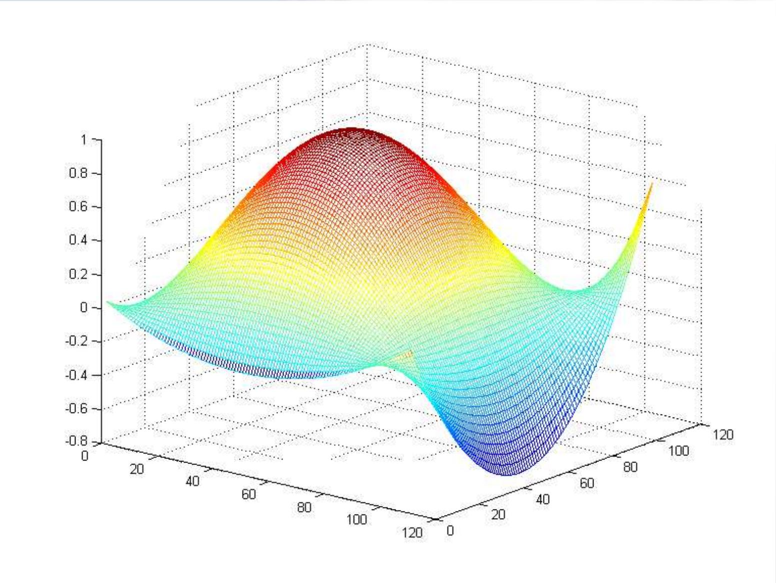
- In most real-life situations, we can’t say for sure how changing the system state will affect our progress toward a goal
- Machine Learning
  - Online Searching
    - Hill Climbing, Simulated Annealing, Genetic Algorithms, Swarm Intelligence
    - Try out different variations of system inputs to see how they affect outputs, until a “best” combination is found
  - Statistical Approaches
    - Neural Networks, Hidden Markov Models, etc
    - Use training data to develop statistical relationships between system inputs and outputs in order to predict behavior

Hill Climbing

- Find *x* and *y* to maximize f(*x*, *y*)
- Function f(.) is unknown
- Trying every possible *x* and *y* is too time consuming
- Start with some initial *x*, *y*
- Each iteration:

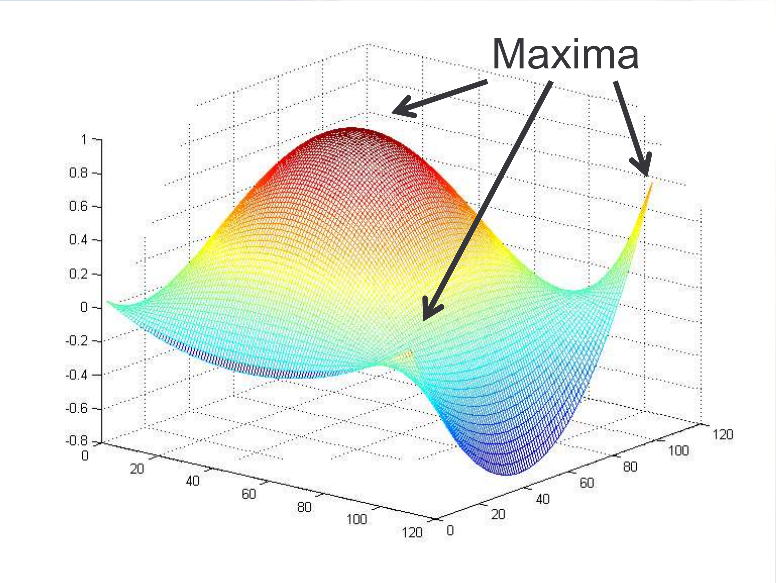


f(x-1, y-1)	f(x, y-1)	f(x+1, y-1)
f(x-1, y)	f(x, y)	f(x+1, y)
f(x-1, y+1)	f(x, y+1)	f(x+1, y+1)



Simulated Annealing

- Hill climbing can yield the wrong result if there are multiple maxima
- “Climb to the top of the smaller hill”
- Simulated annealing does the process many times, each time with a different starting point
- Random perturbations decrease in size as the process executes



Genetic Algorithms

- Encode function parameters as *chromosomes* of *individuals*
- Fitness of each *individual* equals the function evaluation
- *Mutations* and *Reproduction* result in better *individuals*, and the inferior die
- Genetic pool is the set of best candidate solutions

q3 r1 s2 t1 u2 v1 w3 x1 y1 z1

q1 r1 s1 t1 u1 v1 w1 x1 y1 z1

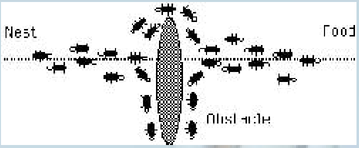
q1 r2 s2 t1 u1 v1 w2 x1 y1 z1

q3 r1 s1 t1 u1 v1 w2 x1 y1 z1

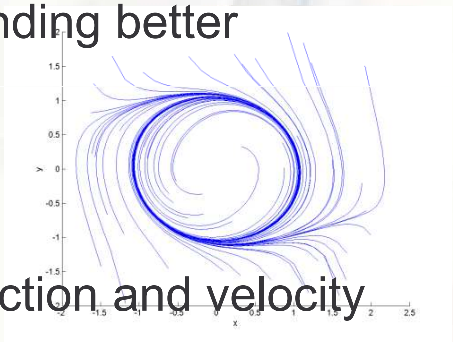
q1 r1 s2 t1 u1 v1 w3 x1 y1 z1

q2 r1 s1 t1 u1 v1 w1 x2 y1 z2

Swarm Intelligence



- Many different logical *agents* climb the same landscape of hills
- **Ant colony optimization**
  - Each agent searches randomly, and upon finding a maxima, deposits pheromone trail while returning "home"
  - Others can follow pheromone trail and build upon it, finding better solutions
- **Particle swarm optimization**
  - Agents can directly communicate
  - Each agent's path is defined partially by their own direction and velocity and also that of others in the swarm
- Various techniques to try and break the problem down – have different agents investigate different dimensions
- Better than simulated annealing if function *f* is changing dynamically



(images from *tecfa.unige.ch* & *inria.fr*)

Neural Networks

- Tries to come up with a linear approximation of f(.)
- Provide many values $\{q_i, \ldots, z_i, f(q_i, \ldots, z_i)\}_I$
- System uses values to train linear approximator
- Can predict value of f(.) for other inputs
- Use training data to compute closest-match α_i
- Simple one-stage neural network:

Learned Relationship

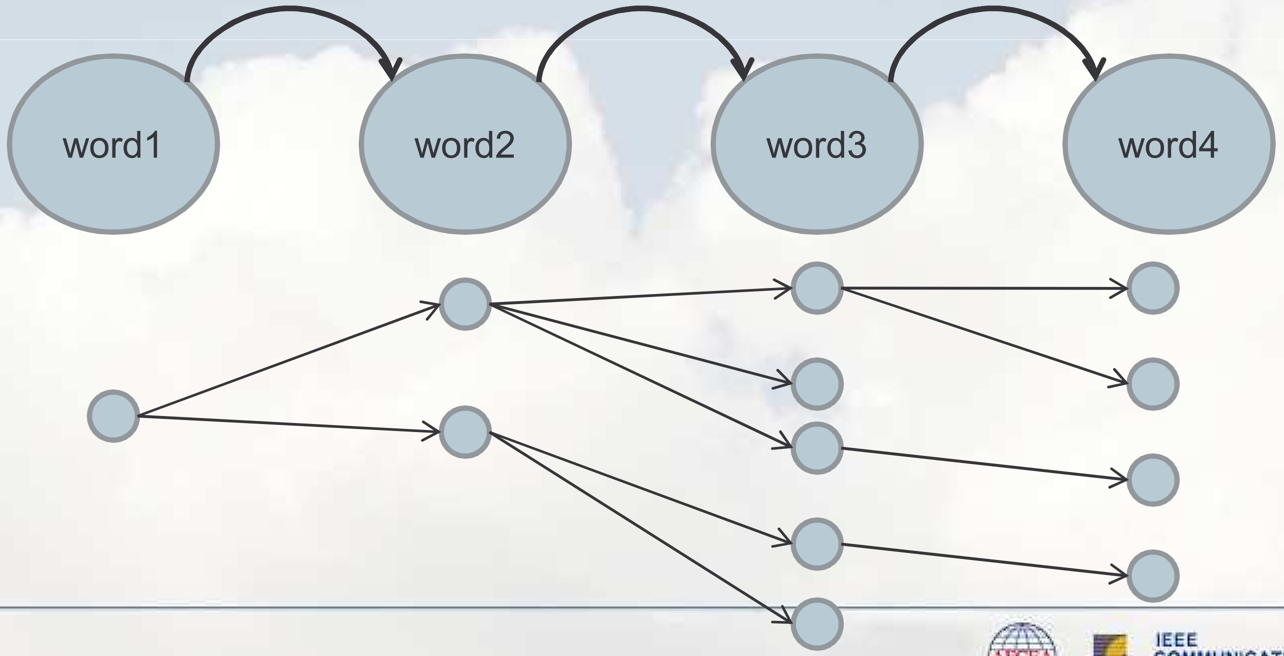
$$\begin{bmatrix} w_i & x_i & y_i & z_i \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = f(w_i, x_i, y_i, z_i)$$

Input Parameters

Output Value

Hidden Markov Models

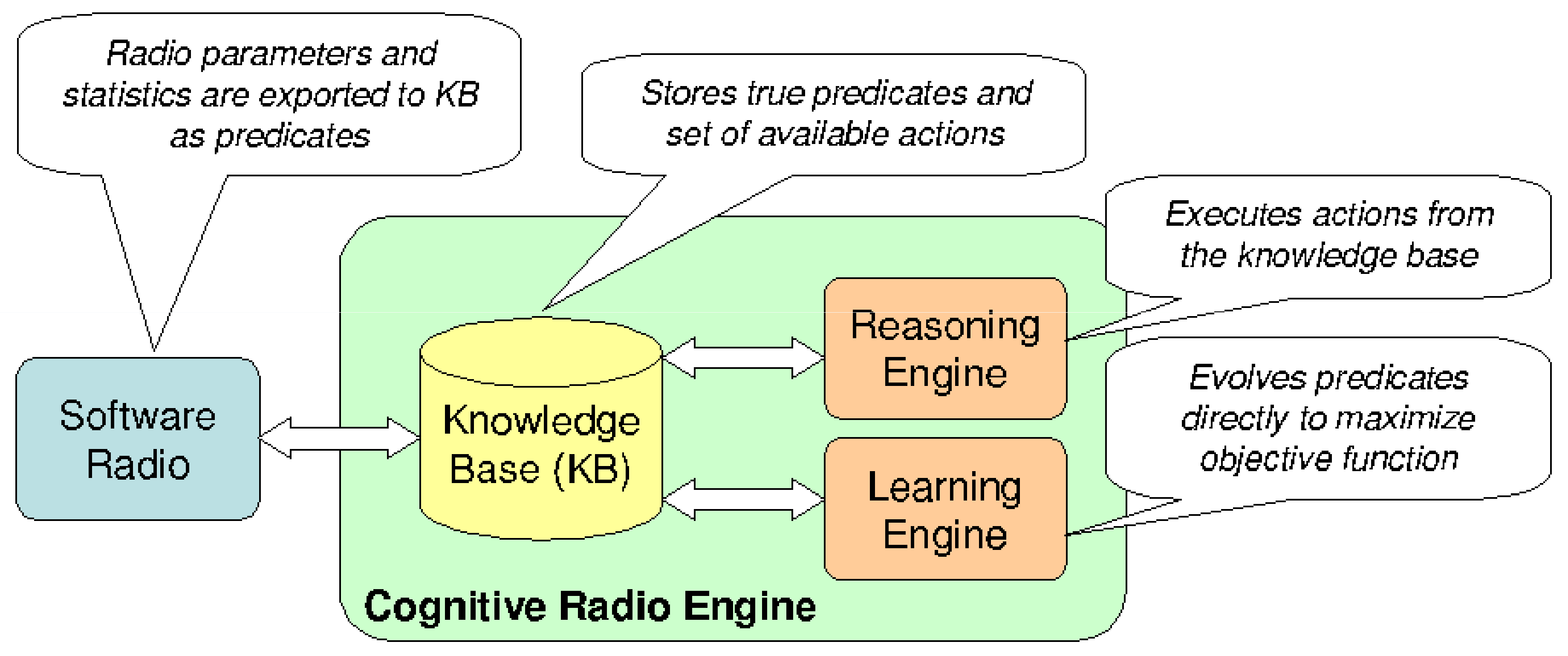
- Probabilistic model looking at ordered inputs
- Major applications in speech recognition
- Compute probability of all possible word$_i$ given observed word$_i$' and word$_{i-1}$ – follow path of highest probability (train using corpus)



Applications to CR

- How do AI techniques apply to CR?
- Policy radios
  - Preprogrammed with a set of rules
  - For every situation, look up in a table to determine action
  - **AI PLANNING**
- Learning radios
  - Can adapt to unforeseen situations
  - Do not require extensive preprogramming – build up knowledge
  - Require learning phase to acquire domain knowledge
  - **AI LEARNING**

Cognitive Radio



Radio parameters and statistics are exported to KB as predicates

Stores true predicates and set of available actions

Executes actions from the knowledge base

Evolves predicates directly to maximize objective function

Cognitive Radio

- Two major types
  - – Policy Radios
  - – Learning Radios
- Two major applications
  - – Adaptive waveforms
  - – Dynamic spectrum access
  - – (and various combinations)

Adaptive Waveforms

- Radios that change their waveform properties to work best in their current environment

- Given a particular objective (computed as a function of inputs and outputs), pick inputs to maximize objective

Policy vs Learning

- When is a Learning radio necessary?
- Policy radios can determine best course of action if objective function is evaluatable on inputs and static outputs
- Learning radio is required if relationship between inputs and outputs is not known
  - Use online searching to find best solution
  - Use statistical technique to understand relationship and allow policy engine to determine best solution

Policy vs Learning Example

- Goal: maximize channel capacity
- Inputs: modulation rate, coding rate
- Outputs: SNR, BER

- SNR independent of inputs
- If noise is AWGN, can find formula relating capacity to modulation and coding rates – **Policy Radio**
- If noise is NOT AWGN, need to use BER (function of modulation and coding) to determine best choice – **Learning Radio**

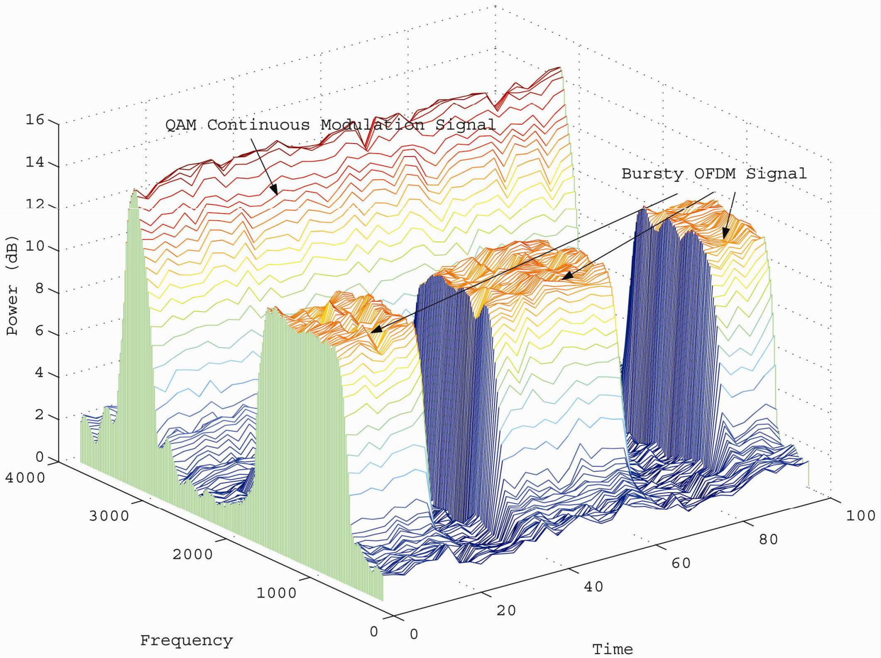
Adaptation Examples

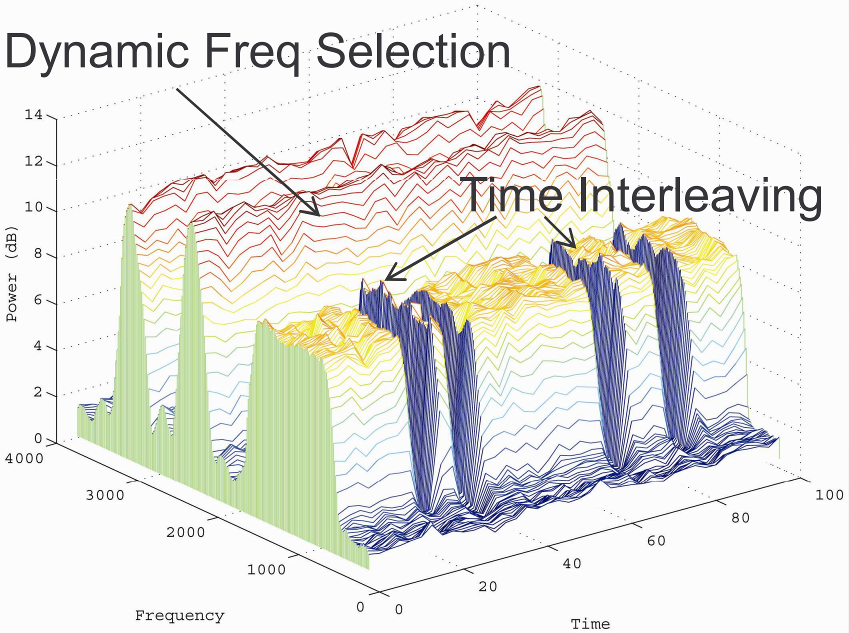
- Adapt to changing noise
- Adapt to changing interference
- Adapt to jamming signal
- Adapt to remaining battery life
- Adapt to changing applications
- Adapt to minimize detection

Dynamic Spectrum Access

- Major commercial application of CR
- Most RF spectrum allocated but unused
- Find idle spectrum and use it to communicate
- Stop using when spectrum owner needs it

Examples




Dynamic Freq Selection

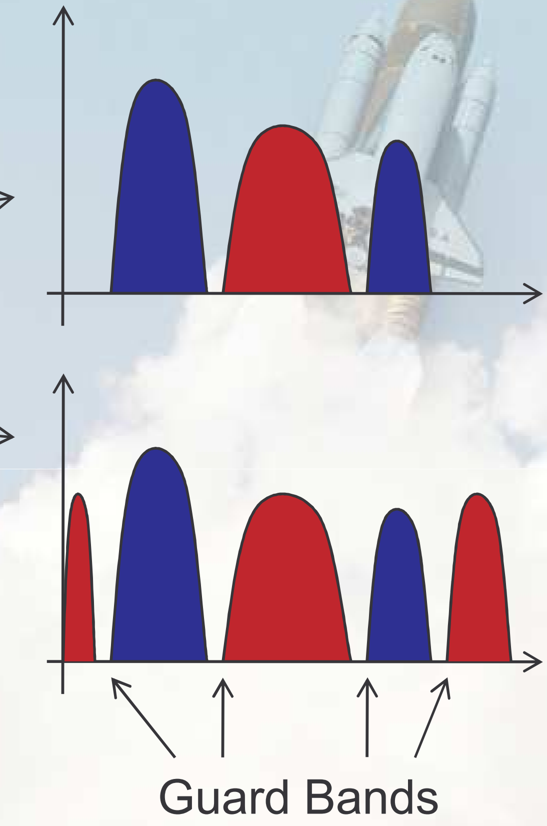
Time Interleaving

Types of Access

- Frequency-Domain Coexitence
  - Unlicensed Radios pick vacant frequencies
  - Spectrum utilization 20% → 80%
- Time-Domain Coexistence
  - Unlicensed Radios detect when channel is idle
  - Spectrum utilization 80% → 95%
- Interference-Domain Coexistence
  - Careful power control, transmit at same time and frequency as licensed users
  - Spectrum utilization 95% → 99%

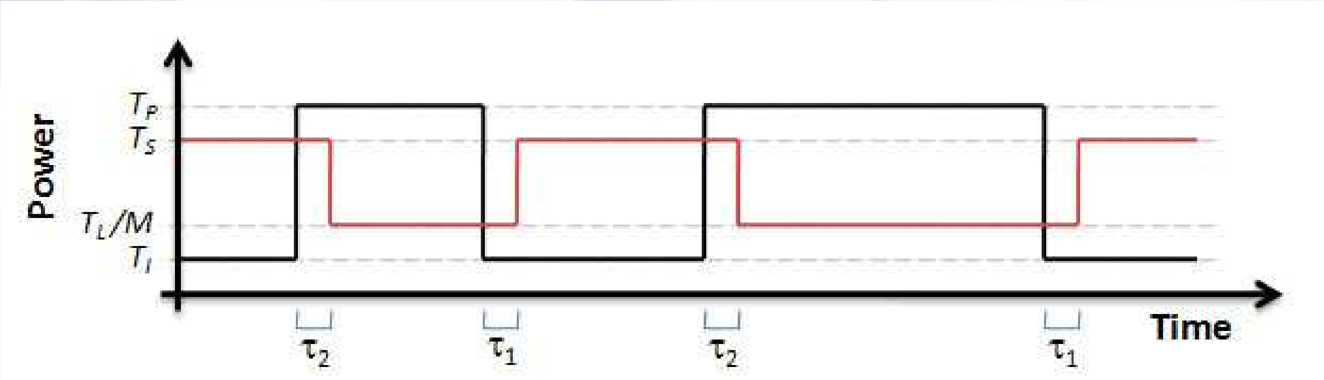
Frequency-Domain

- DARPA XG Project
- Two types of radios
  - Simple Radios
  - Pooling Radios



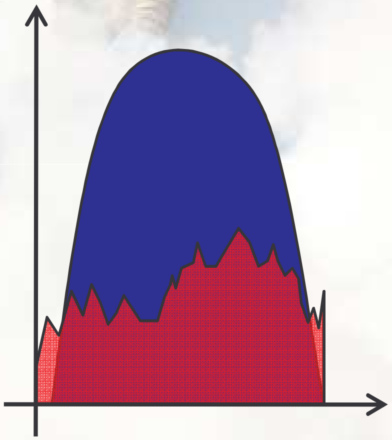
Time-Domain Coexistence

- Sense spectrum to determine when primary user is idle
- Start transmitting
- Stop when the primary user is detected



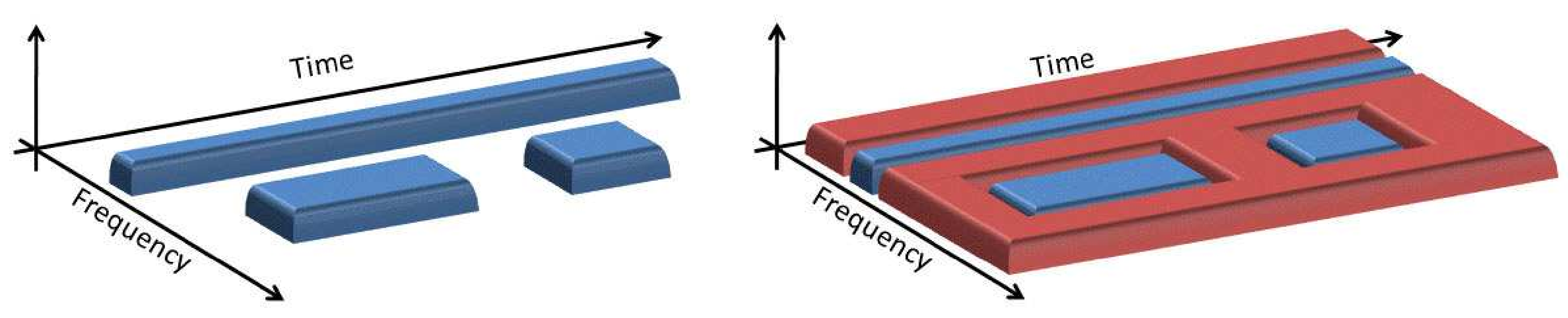
Interference-Domain

- Transmit at same frequency and time as active signal
- Power is governed by Interference Temperature Model
- Interference
  - Raising the noise floor
  - Inaccurate guess as to distance between secondary transmitter and primary receiver



Hybrid Approaches

- Combine multiple approaches
  - Frequency-Domain
  - Time-Domain
  - Interference-Domain



AI in DSA

- Signal detection algorithms
  - Neural networks
- Primary user access patterns
  - Hidden Markov models
- Determining optimal freq/time accesses
  - Planning

CR Projects

- DARPA
  - – XG (neXt Generation) Communications
    - Frequency-domain dynamic spectrum access
    - Next-generation JTRS waveforms
  - – WNaN (Wireless Network After Next)
  - – WANN (Wireless Adaptive Network Node)
  - – WAND (WNaN Adaptive Network Development)
    - “develop and demonstrate technologies and system concepts that will enable intelligent adaptive wireless networks consisting of densely deployed low cost wireless nodes”
- IEEE 802.22
  - – Standard for MAC/PHY with DSA support

CR Projects

- Generic CR Platforms
  - Virginia Tech
  - Laboratory for Telecommunications Sciences
  - University of Kansas
- Seek to use generic AI constructs that allow a radio to be programmed for any objective

Conclusion

- Multiband cell phones using SDR-like technologies
- Commercial GSM base station in SDR (Vanu)
- Within the next year CR technology will be commercial (mostly DSA)
- We already see some adaption
  - WiFi, WiMAX
  - power control, adaptive modulation/coding
  - often suboptimal because it’s not learning-based
- Future radios may not even need standards; networks would make them up as they go