A Design Workflow for Software Defined Radios

Frederick Weidling
tedjw@itc.ku.edu

Presented August 22, 2007 to:
Dr. Gary J. Minden
Dr. Joseph B. Evans
Dr. Alexander M. Wyglinski
Acknowledgements

• Defense Committee
  • Dr. Gary J. Minden (Chair)
  • Dr. Joseph B. Evans
  • Dr. Alexander M. Wyglinski

• KU Agile Radio Group
  • Leon Searl and Dan DePardo
  • Jordan Guffey, Rory Petty, Tim Newman, Brian Cordill, Dinesh Datla, Rakesh Rajbanshi, Qi Chen, and Megan Peck
Outline

• Background
  • Software Defined Radios (SDR)
  • Dynamic Spectrum Access (DSA)

• Proposed Research
  • A workflow for SDRs

• Validation
  • Apply design workflow to KUAR
  • Generate systems

• Extension
  • Generic SDR API
  • Hardware Agnostic Cognitive Network

• Conclusion
What is an SDR?

• **Ideal SDR**
  - Direct conversion between digital samples and analog waveform
  - Use only general purpose processors (GPP)
  - Unrealistic for high frequencies – cost, processor speed, algorithm complexity

• **Practical SDR**
  - Baseband processing, analog components translate to transmission band
  - Optimizations using digital signal processors (DSP) and field programmable gate arrays (FPGA)
Why do we need SDRs?

- FCC “Command-and-Control” Policy Limits
  - What can be transmitted
  - Who can transmit
  - Where they can transmit
- Results in apparent spectrum scarcity
  - Restricts research and novel services
  - New York 13.1% utilization 3.0 MHz – 3.0 GHz
  - TV utilization often less than 50%
- Solution: SDRs
Dynamic Spectrum Access

- SDR detects unused frequencies (whitespace) and utilizes them
- Cognitive Radio (CR)
  - Radio which adapts to its environment
  - Cognitive algorithms implemented on SDR platforms
- Dynamic Spectrum Access (DSA)
  - Avoid licensed users
  - Utilize whitespace

Current SDR Technologies

• Universal Serial Radio Peripheral (USRP)
  • Generic RF front-end
  • Modular transmit/receive bands

• GNU Radio
  • Open source SDR software
  • Only supports general purpose processors
  • Based around USRP

• Global Mobile (GloMo) API
  • Generic radio modem API

• Software Communications Architecture (SCA)
  • CORBA based communications model for SDR modules

• Joint Tactical Radio System (JTRS)
  • SDR platform commissioned by Defense Advanced Research Projects Agency (DARPA)
  • Built using SCA
The KU Agile Radio (KUAR)

- Digital Board
  - 1.4 GHz Pentium M
  - 1 GB RAM & 8 GB microdrive
  - PCI Express, USB, 1 Gb ethernet
  - 160 MSPS Digital-to-Analog Converter (DAC)
  - 105 MSPS Analog-to-Digital Converter (ADC)
  - Xilinx Virtex II Pro 30 FPGA

- RF Front-End
  - Modular
  - 5.25 GHz – 5.85 GHz UNII Research Band
  - Receiver sensitivity -100 dBm
  - Transmitter power +25 dBm
  - Quadrature modulation & demodulation
  - Baseband bandwidth of 30 MHz
SDR: A Federation of Components

• Variety of components must work in unison to transmit/receive
  • General purpose processors
  • Special purpose processors (FPGA, Microcontrollers)
  • Frequency synthesizers
  • Modulators/Demodulators
  • Attenuators

• Different types of design problems on SDR
  • Cognitive network design
  • Real time data management
  • Communication systems
Development Domains

• Break the problem into domains
  • Reconfigurable hardware domain aimed at Communications Engineers
  • Embedded software domain aimed at System Engineers
  • Radio management domain aimed at Network Engineers and general radio maintenance

• Each domain has its own workflow requirements
  • Different set of development tools
  • Each domain’s toolset must support designing, implementing, and verifying modules
  • Domains interact but implementations are independent
Reconfigurable Hardware Domain

• Implement physical layer communication systems
  • Direct access to ADC/DAC
  • Signal processing optimizations possible in FPGA

• Generic memory elements need to be provided
  • Bus controller
  • Buffers (FIFOs, RAMs)
  • Registers (Status, Control)
Embedded Software Domain

- Manage hardware interfaces and data streams
- Components
  - Dynamic Hardware Interface: configure/communicate with reconfigurable hardware
  - Static Hardware Interface: communicate with RF modules and sensors
  - Spectrum Sensing: perform spectral sweeps
  - Waveform Protocols: physical layer protocols
Radio Management Domain

- Radio Management
  - Diagnostic tools
  - Network protocols
  - Spectrum access protocols
- User interface enables network tests
  - Control multiple radio nodes
  - Define co-operative tests
  - Display results
Implementation on the KUAR

- Reconfigurable Hardware Domain
  - Xilinx Virtex II FPGA
  - PCI, PCIe, or USB bus
- Embedded Software Domain
  - 1.4 GHz Pentium M with 1 GB RAM
- Radio Management Domain
  - Remote KUAR interface
Communication Systems

- Development tools
  - Develop using Matlab/Simulink
  - Implement using Xilinx VHDL
  - Verify with Simulink, Modelsim, and hardware test bench
- Test cases translated down flow
- Verified components added to shared library
- Systems Built
  - BPSK, MQAM, OFDM, Spectrum Analyzer, …
Support Libraries

- Development tools
  - Implement using GCC & GNU Makefile
  - User validation & cUnit
- FPGA Library
  - Configure bit file
  - Read from and write to hardware configurations
- RF Control Library
  - Manage the RF front-end
  - Set and get frequencies and gains
  - Query hardware capabilities
- Monitor Library
  - Monitor system temperatures
The KUAR Control Panel

- Control multiple radios from one interface
  - Radio status
  - Configure radio profile
  - Control RF front-end
- Define network and single radio tests
  - Test definitions XML based
- Extendable interface
  - Java based remote KUAR API
  - Programmable test window
KUAR Control Panel: Visualizations

Constellation for QPSK Transmitter

Eye-Diagram for QPSK Transmitter

Spectral Response of a Modulated Signal Using a Flat Filter

Whitespace Detector for an OFDM Transmitted Signal
Achievements

• Developed design workflows to handle
  • Reconfigurable hardware
  • Embedded software
  • Radio management

• Implemented on the KUAR
  • Communications systems
    – BPSK, QPSK, OFDM
  • KUAR API
    – RF Control, FPGA Control, Monitoring libraries
  • Analytical systems
    – Whitespace detector, spectrum analyzer
Enabling Generic SDR Development

• Physical layer implementations often tailored to system
  • Dependent on sampling rates
  • Optimized for parallel and signal processing
  • Hard real-time deadlines

• Cognitive algorithms are complex but more generic
  • Wide variety of issues: whitespace, battery life, channel characteristics, noise power, …
  • Complex solutions: frequency selective modulation, expert systems, genetic algorithms, ontology based systems, …

• Should be able to re-use cognitive algorithms
Hardware Agnostic Network Stack

- SDR Platform API handles the physical layer implementations
- Unifying Layer exposes generic SDR interface
- Traffic Scheduler manages access to shared RF front-end
- Cognitive Network Layer is independent of SDR implementation platform
Unifying Layer

- **Hardware Manager**
  - List hardware capabilities
  - Control RF front-end
- **Protocol Manager**
  - List waveform protocols
  - Manage waveform protocols
- **Spectrum Sensor**
  - Handle spectral sweeps
- **Data Stream Manager**
  - Similar to GloMo API
  - Interact with configured waveform protocols
Traffic Scheduler

- Schedule packets to channels
  - Channel is a hardware digital/analog chain
  - Scheduled packet includes protocol, frequency, power, and time frame
  - Each channel has an allocation list

- Packets are scheduled in a sliding window
  - Packets may be scheduled periodic or single-shot
  - Call back interface for error states and packet completion
Traffic Scheduler: State Diagram

- **Start Traffic Scheduler**
- **Idle**
  - Schedule packet request
  - No more scheduled packets
- **Configure Profile**
  - Scheduled packet start time
- **Schedule Packet**
  - Wait for next scheduled packet
- **Wait**
  - Schedule packet request
  - Wait for next scheduled packet

The state diagram illustrates the lifecycle of traffic scheduling with transitions between states.
Conclusion

• Achievements
  • Developed a design workflow for SDRs
  • Workflow used to develop systems for KUAR
  • Proposed generic SDR API (Unifying Layer)
  • Described infrastructure to support hardware agnostic cognitive networks

• Future work
  • Implement hardware agnostic network stack
  • Develop more robust cognitive algorithms for KUAR
  • Apply workflow to another SDR platform
Resources

- **SDR Resources**
  - SDR Forum: http://www.sdrforum.org/

- **Publications**

- **Questions?**