Cognitive Radio Platform Research at WINLAB

December 2, 2010

Zoran Miljanic and Ivan Seskar
WINLAB
Rutgers University

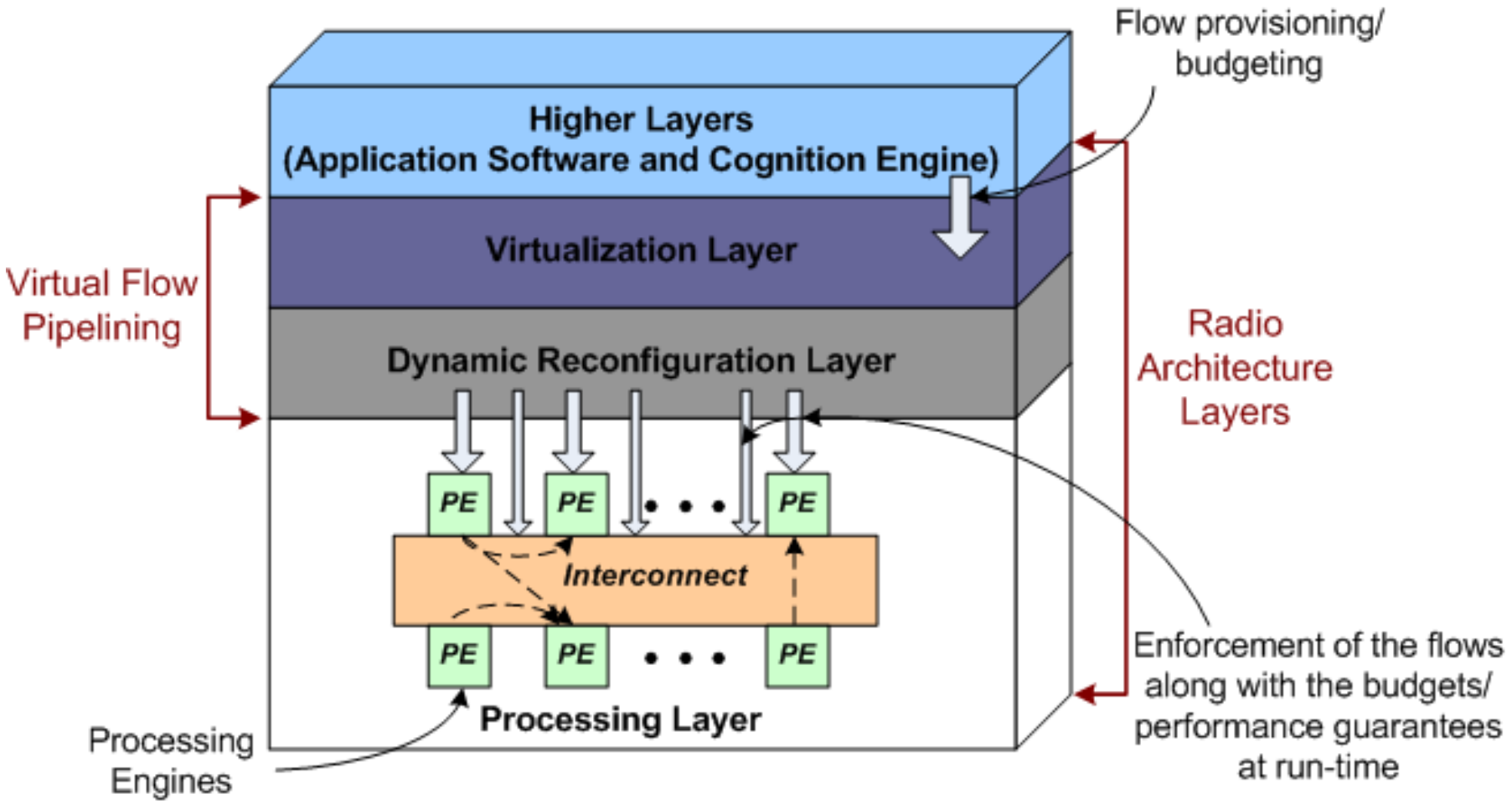
www.winlab.rutgers.edu

WINLAB
WIRELESS INFORMATION NETWORK LABORATORY

WiNC2R objectives

- Programmable processing of phy and higher layer at speed – target rate 500 Mbps
- Deploying processing engines to satisfy **functionality and performance** needs of the CR environment
- Event driven execution model with deterministic and best effort performance guaranties. Processing is a sequence of synchronous and/or asynchronous tasks
- Programming at two levels:
  - System level: combining the operations of built-in functional modules to satisfy the protocol, performance and time constraints.
  - Programmable CPU-s allow for on-the-fly architectural modification; SW defined modules plug into the processing flow as any other functional unit
- Control mechanisms for application design and system integration:
  - controlled sharing of the resources among applications
  - preserving guaranties for individual applications

Processing paradigm



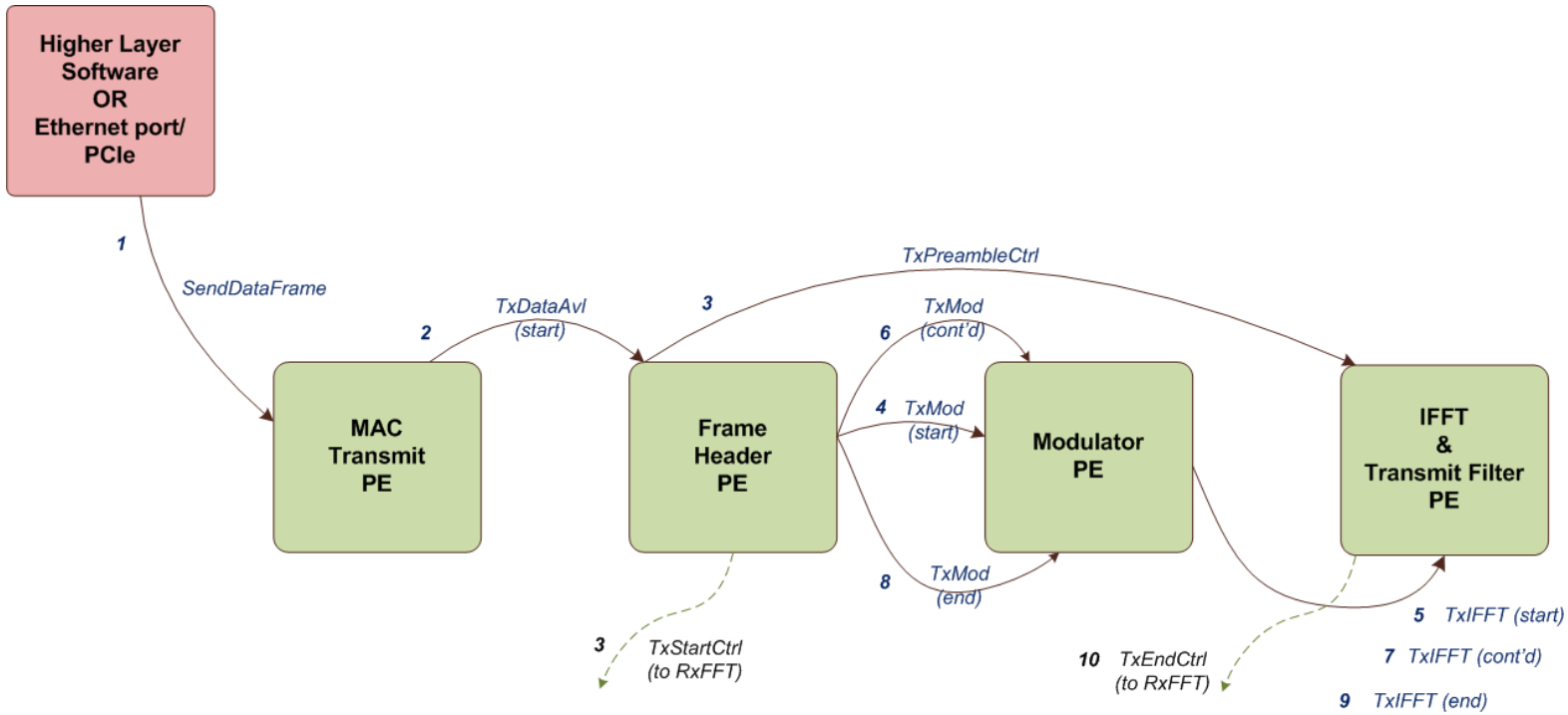
Programming mechanisms

- The application is designed by programming data processor threads for the new functions, and combining functional unit commands and data processor threads into the communicating tasks
- The following classes of design and run time control mechanisms are provided:
  - Task sequencing: orders tasks into the application. Sequencing patterns supported: one-to-one; one-to-many, many-to-one, many-to-many
  - Task synchronization: resolved producer-consumer relationships for all sequencing patterns
  - Task repetition
  - Scheduling: synchronous and multi priority asynchronous
  - Communication between the tasks
- All of these mechanisms are supported in hardware by the uniform Unit Control Modules – every functional module has the same controller that communicates with the controllers of the other units as per program design
- The program and system control is designed through the set of data structures:
  - Global Task Table
  - Flow Descriptors and Application DAG
  - Scheduler Queues
- New functions designed as assembly or C/C++ program of data processors

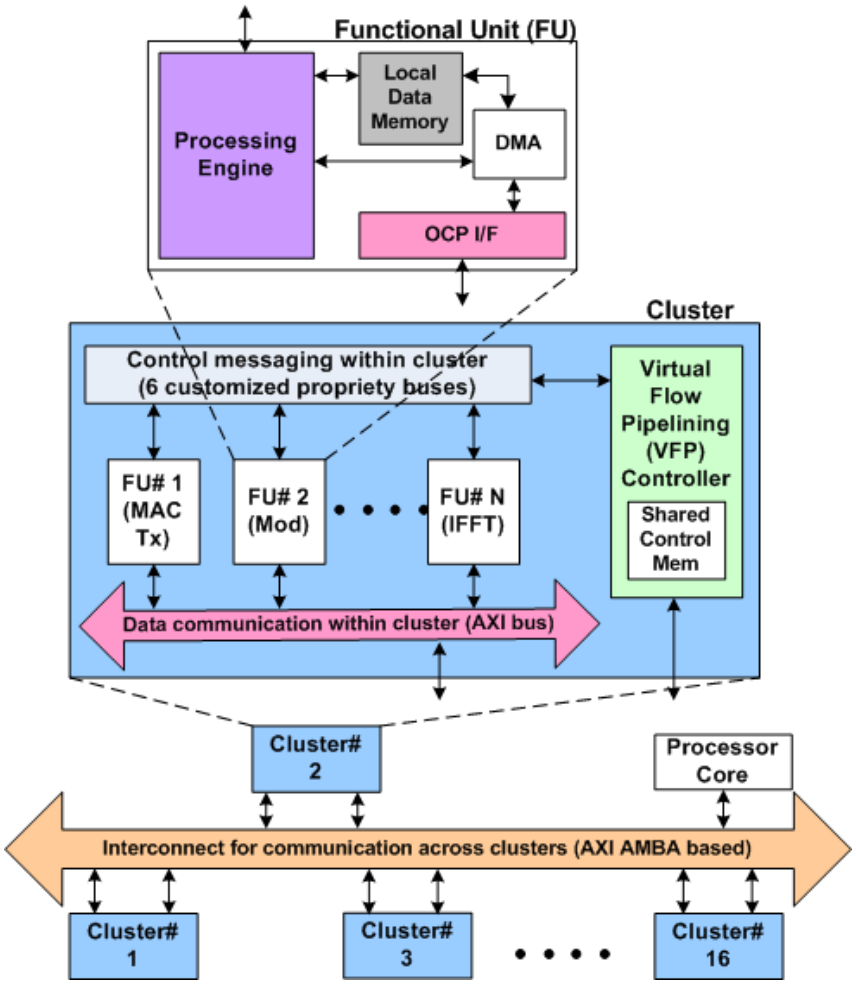
Virtual Flow Pipelining

- Virtual Flow Pipelining (VFP) accommodates functional, time constraints and performance requirements of communication protocols (phy and higher) while allowing flexibility in the following dimensions:
  - Selecting the operation at each stage of processing from a predefined set
  - Option of software defined function at any stage while respecting the pipelining I/O and processing latency constraints. Data processors under system pipeline control, rather than a CPU controlling the pipelining. Simple processor, but requires separate control structure and hardware scheduler
  - Selecting the order of execution of the operation in the processing stages
  - Enforcing task flow end-to-end and repetition interval time constraints per virtual flow
  - Coexistence of multiple virtual flows
- The Virtual Flow Pipelining is the task sequence described with the following relationships and properties:
  - Relationships: consumer-producer, synchronization, communication
  - Properties: task actions, task/flow priorities, repetition interval

Transmit VFP task sequence



WiNC2R Cluster Based Architecture



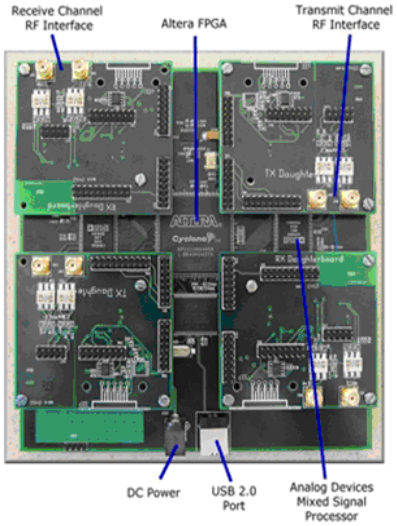
- Cluster-based design – improves scalability
- Hierarchical interconnect – improves on-chip communication bandwidth and latency
- VFP controller – orchestrates the Virtual Flow Pipelining – acts as a common, shared Unit Control Module across FUs

Project Status

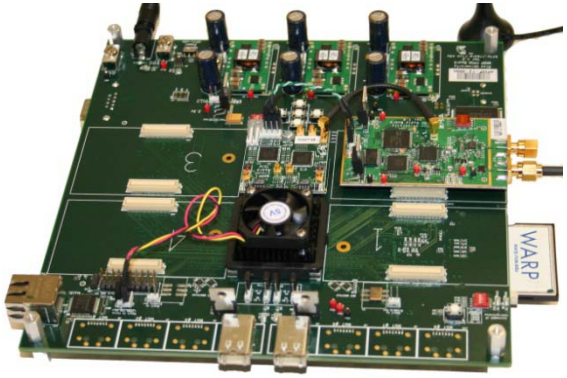
- Build FPGA proof of concept OFDM transceiver prototype:
  - Uses one Xilinx XC Virtex-5 SX95T for transmitter and one SX95T device for receiver
  - Based on Virtual Flow Pipeline architecture
  - Input: 125 MSps, 14; bits Output: 500 MSps
  - RF bandwidth 40 MHz, frequency range programmable to: 2400–2500, 4900– 5300, 5400–5875 MHz
- Build virtual prototype - foundation for ASIC development
  - AXI based SoC infrastructure
  - System Verilog OVM based architecture evaluation and functional verification completed
  - Cluster based scalable architecture design completed
  - Integrated Tenslica based Application Specific Instruction Set Processors (ASIPs) with support for multi-standard interleaver/de-interleaver and MIMO detection

Prototype design

ORBIT Cognitive “Capable” Platforms

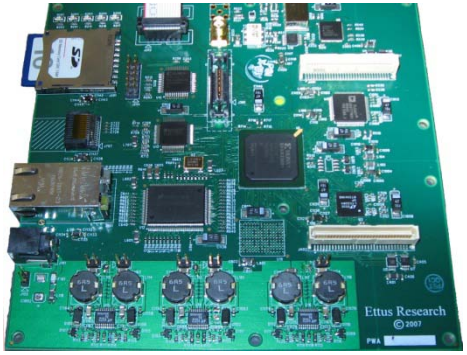


USRP



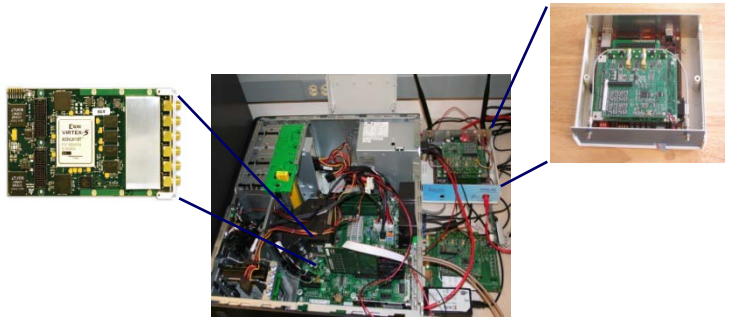
RICE WARP Platform

U. Of Colorado



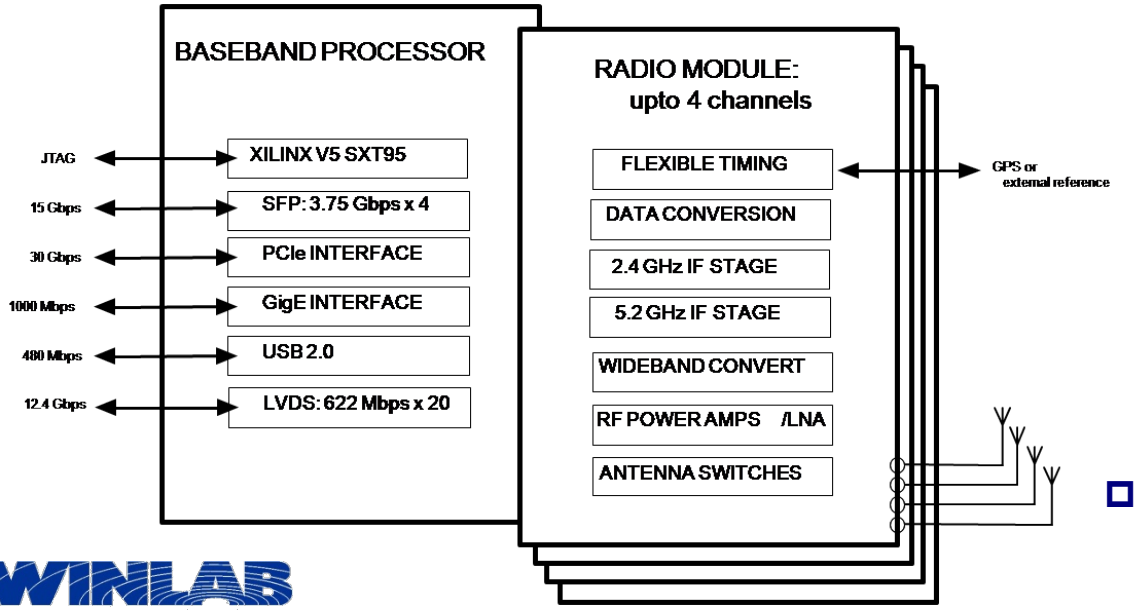
USRP2

GENI CRKit



WINLAB WINC2R System

GENI Cognitive Radio Kit (CRKit)

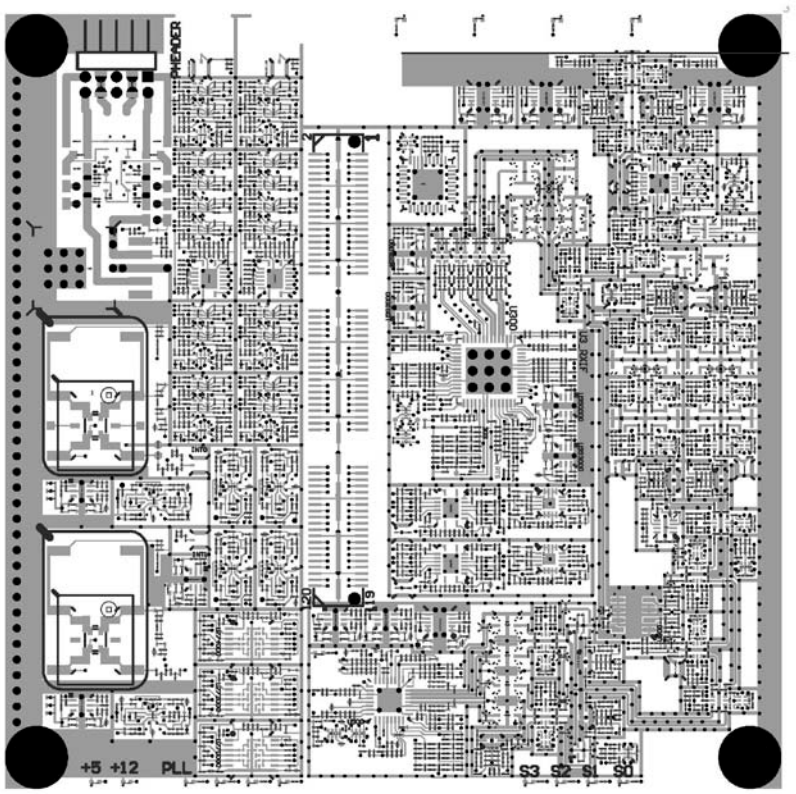


Open Source Platform

- **Range of COTS baseband FPGA platforms**
  - Medium size (LX50)
  - Large size (SX95)
  - **Standard interfaces:**
    - **1000 BaseT, (SFP)**
      - **USB**
    - **(8x PCIExpress)**
  - **4 (2) configurable radio modules for phased or smart antenna applications:**
    - **SDR/F – 25 MHz, 100 M - 2.4/5GHz**
    - **WDR – 25 MHz, 100MHz -7.5GHz**
    - **XDR – 500 MHz, 100MHz-7.5GHz**
- **Application framework with support for both RTL and Matlab (Simulink)**

WINLAB
WIRELESS INFORMATION NETWORK LABORATORY

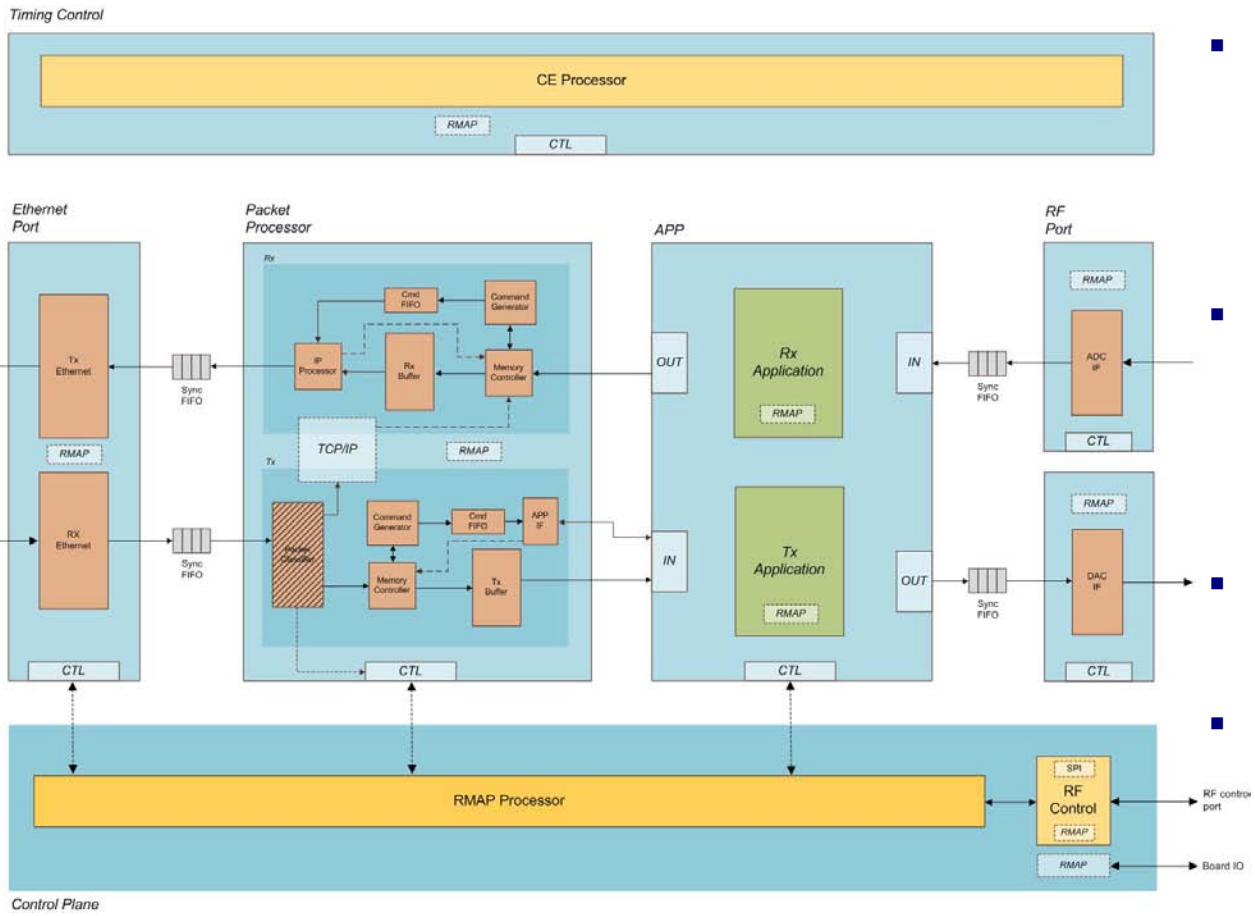
WDR RF Front-end (Curr: WDR v2.02)



- **14 layer PCB with high-frequency 5.5 mil thick NELCO N4000-13 material**
- **6000 part footprints with more than 4800 parts**

- Full duplex operation.
- One to four independent radio modules on one (FPGA) processor.
- Each module allows two up to 40 MHz bands from 100 to 7500 MHz
  - 12 bit ADC sampling up to 80MSps on both I and Q rails.
  - NF = 6dB, optional external LNA for customized applications.
  - 70dB of RX gain control.
  - 14 bit DAC sampling upto 200MSps on both I and Q rails.
  - +20dBm TX output power with fast gain control.
  - 60 dB of TX gain control
- 1 usec RF frequency switching time
- Switched antenna diversity for both TX and RX channels
- Comprehensive reference clock selection or generation with internal, external or digitally derived sample clocks.
- Multiple infrastructure-class data pipes.
- Extensive built-in-test for monitoring system status and health (including loopback).

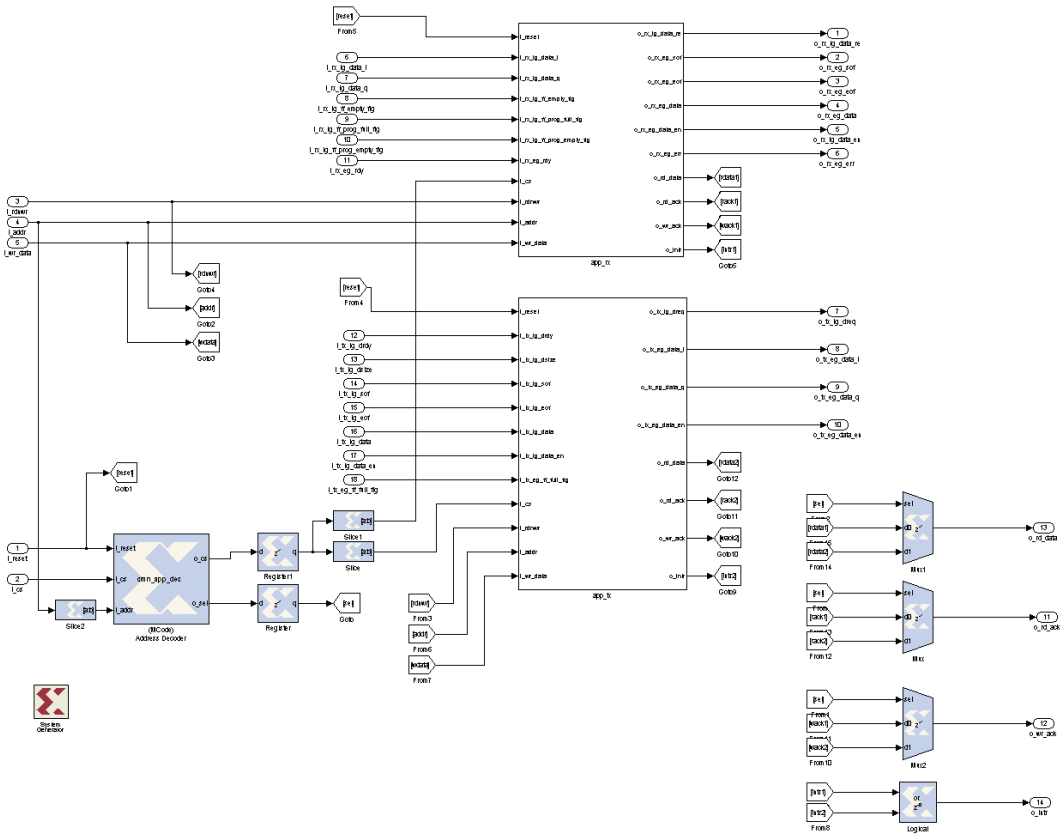
GENI CRKit Framework



Feature :

- Fully functional FPGA development platform with Pluggable User Apps .
- Two environments: MATLAB/Simulink or VHDL/Verilog/BlueSpec
- Communication with Host using GbE links
- Streamlined FPGA building process e.g. HW design made-easy using MATLAB/Simulink and build scripts.

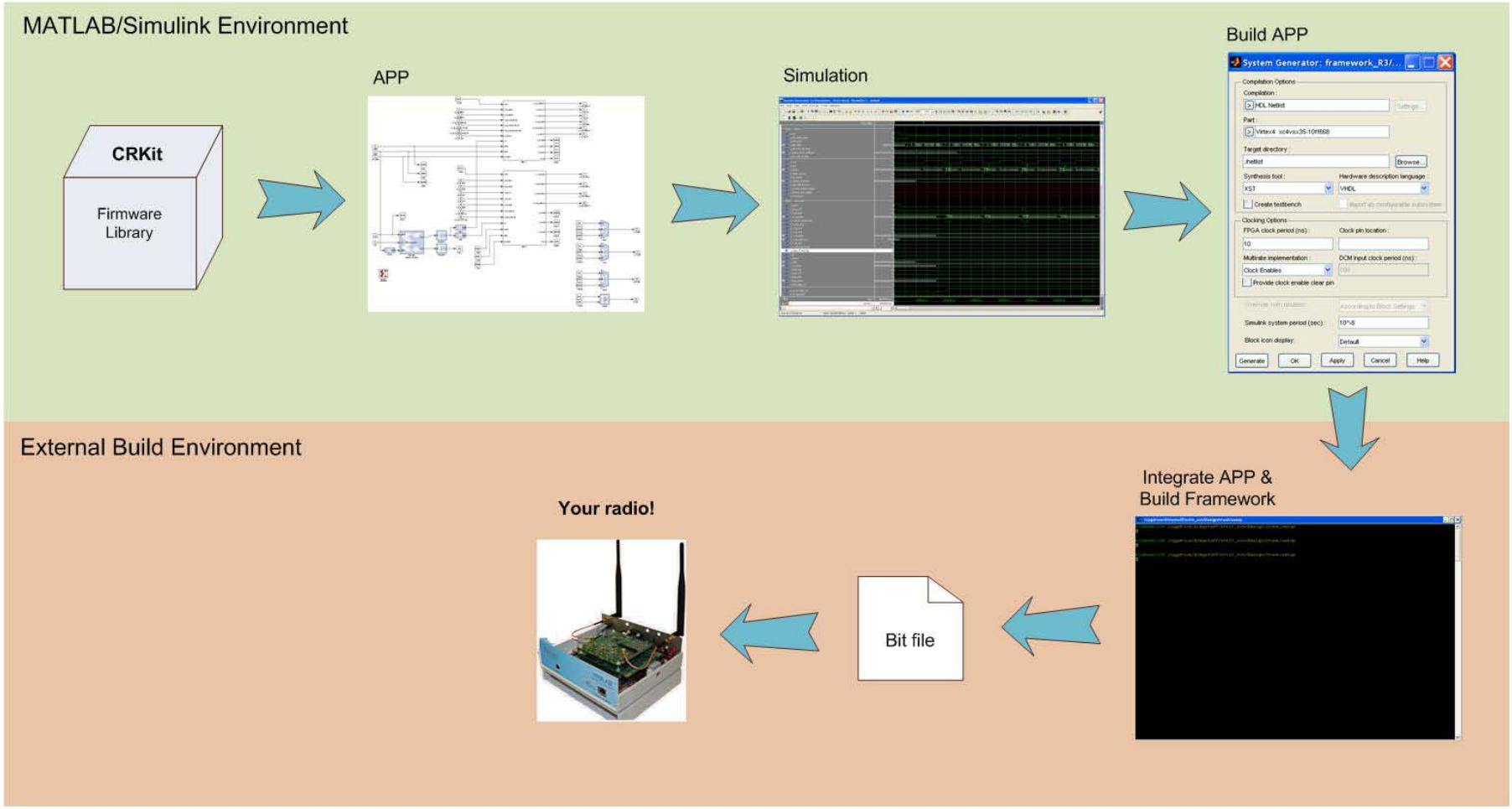
APP Development



APP subsystem:

- Build App as separate entity in MATLAB/Simulink, then integrate into Framework.
- Integration of Tx and Rx subsystems
- Well-defined IO interfaces between APP and Framework.
- Either develop own Tx/Rx Apps, or select from libraries

Framework Design Flow



Issues

- Main issue to date is the use of licensed software (namely Xilinx and Matlab) that is required for FPGA code development. This is especially troublesome for deployments in remotely accessible testbeds not only because it requires that each user have a valid license, but because most academic licenses require that the actual development be done on the machines that belong to the licensee (rather than on the infrastructure machines that the testbed provides). Same issue exists for other FPGA based solutions (like NetFPGA)
- Specification tools like BlueSpec reduce some of these issues (matlab/simulink), but don't get around basic Xilinx license issue
- Outreach to Xilinx Research at Colorado to explore options

CRKit Future Plans

- Conversion of waveform to more modular form (BlueSpec)
- Validation of waveform control across wide-band front-end
- Further development of basic communication blocks (GENI Radio Library)
- Phase 2 radio development (wider baseband)
- Multi-FPGA support and design partitioning
- Wider GENI deployment