Software Defined Radio (SDR) Architecture and Systems Issues

Workshop on
Spacecraft Flight Software (FSW-09)
2009-11-6

Kenneth J. Peters
Jet Propulsion Laboratory, California Institute of Technology
Introduction

• Speaker
• NASA STRS
  – Space Telecommunications Radio System
• JPL Implementation
• Systems Issues

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

Speaker

- JPL software engineer for 15+ years
  - Software lead for Electra radio software for MRO and MSL
- Helped draft and review NASA STRS specification
  - Spec is released, but improvement is ongoing
- One of several software engineers developing the JPL implementation
  - JPL implementation is one of several to help shake out the specification
• Combined effort by several NASA centers with input from industry
• Released (v1.02) hardware, FPGA, and software architecture standard
  – Intended to allow cross-platform portability of communication implementations
  – All layers and signal processing
• Leans toward requiring interface documents, rather than specific interfaces
  – Except for software
Conceptual Block Diagram

General Purpose Processing Module
- CPU(s)
- Memory (volatile and non-volatile)
- System bus
- Spacecraft command/control interfaces

Signal Processing Module
- FPGAs, ASICs, DSPs, etc.
- Memory (volatile and non-volatile)
- Interconnect bus or fabric
- Spacecraft data interfaces

Radio Frequency Module
- Switches, amplifiers, A/D, D/A, etc.
- Waveguides, etc.
- Antennas and/or antenna interfaces
Basic STRS SW Block Diagram

- **STRS API**
  - Control and interaction
  - Similar to
    - OMG/SWRADIO
    - OMG/SCA (JTRS)
  - Generally lighter weight
- **POSIX RTOS**
  - or compatibility layer
- **Device drivers**
  - POSIX and/or STRS
STRS Applications

• The CPU and FPGA software that makes the box “be a radio”

• Intended to be portable across compatible platforms
  – Use STRS and POSIX APIs
  – Use services provided by platform vendor through STRS APIs
    • Such as spacecraft interface handlers (commands and telemetry)
  – Access hardware via device drivers
    • May need to develop compatible drivers on different platforms
  – Use defined and flexible interfaces to and within FPGAs
JPL Implementation

- Basic architectural features run on Linux platform
  - Allows easier development
  - Demonstrates basic operating system portability
- Targeted to custom radio hardware for full implementation
  - Atmel AT697 Sparc CPU
  - Using the RTEMS POSIX compliant RTOS
    - Open source
    - Supports AT697 CPU
  - Developing custom RTEMS POSIX device drivers
  - Developing custom FPGA modules for abstraction
JPL Implementation

• Sticking to “plain C” except where necessary to interact with C++ STRS applications
  – Nod to highly conservative space coding environment
  – Want to support C++ applications developed by others

• Applications statically linked with environment
  – Registration functions for C++ object or C entry points
  – Application startup and control via STRS API function commands
JPL Implementation

• Developing a simple modular FPGA methodology
  – Assign module ID codes for different modules that may be combined into an FPGA bitfile
    • Module ID implies functionality and interface specification
    • Also have version codes for updates
  – Define register base addresses for FPGA modules
    • Defined when modules are brought together into a bitfile
    • Each module internally defines whatever registers it wants
  – Make base addresses and module ID codes available to software in registers at the known base address of the entire FPGA
Systems Issues

• The changeable nature of the software defined radio may lead to changing concepts of how the local spacecraft, the ground, and remote spacecraft interact with radios

• Possible applicability to systems issues with more complex instruments in general
• Boundaries/interfaces likely to be in different places than in traditional radios
  – Conceptual block diagram divisions may not match physical divisions
  – Different designers at different layers tend to share physical resources

• Performance specifications also break apart in different ways
  – Need to clearly define performance at each conceptual layer, which may be developed independently
  – Rather than end-to-end with assumption of single coordinated decomposition
CPU/FPGA Interaction

• It is expected that both the CPU software and the FPGA bitfile may change to provide different radio functions

• Clear and flexible interface definitions are needed
  – Both between and within CPU and FPGA code
  – So that multiple programmers of both CPU and FPGA code can work independently and portably

• Clear and flexible version identification and control are needed
  – So that mismatched software and bitfiles can be avoided
Command/Control Complexity

“I want a radio, not a relationship” – Glenn Reeves

• SDRs tend to have more commands and modes than traditional radios

• Command set, and telemetry formats, may vary depending on what software is loaded
  – And may differ from radio “safe mode” if any

• This may be a problem for traditional spacecraft and ground command/control methods
  – May need coordinated “software defined” interfaces throughout the system, or more standardized interfaces
  – Or SDRs may need to have more internal autonomy
• Not a conclusion, an ongoing exploration
• May need new models for the spacecraft bus protocol
  ¿ More like “plug and play”, with driver discovery as in PCI?
  ¿ More like managing network hosts, with auto configuration as in SNMP?
  ¿ More like net services or service oriented architecture, with discoverable interfaces as in WSDL? (Web Services Description Language)
  ¿ ?

Ken Peters
Workshop on Spacecraft Flight Software
(FSW-09) 2009-11-6