

Facilitating Third Party Waveform Porting on Spaceborne SDR Platforms

2 Dec 2010

Prepared by:

Jerry Brand, Fenton McDonald and Alan Mast
Melbourne, FL

NON-EXPORT CONTROLLED

THESE ITEM(S) / DATA HAVE BEEN REVIEWED IN ACCORDANCE
WITH THE INTERNATIONAL TRAFFIC IN ARMS REGULATIONS
(ITAR), 22 CFR PART 120.11, AND THE EXPORT ADMINISTRATION
REGULATIONS (EAR), 15 CFR 734(3)(b)(3), AND MAY BE
RELEASED WITHOUT EXPORT RESTRICTIONS.

SDR'10

- Why enable 3rd party waveform porting?
- How does porting for space differ from conventional porting?
- Should interface standards be used for waveform porting?
- How can we best manage (re: software control) SDR ported waveforms?
- Does the Operating Environment drive porting?
- Do toolkits assist hardware abstractions?
- Can we pull these ideas together and deliver 3rd party ported waveforms for spaceborne platforms?

- During the development of the Space Qualified SDR, it became apparent that there is value in having a radio anyone can program.
 - This value is achieved not just by writing an application that runs on the General Purpose Processor (GPP) but also by developing the Field Programmable Gate Array (FPGA) code for the signal processing elements.
 - To accommodate the user's desire to develop FPGA applications, Harris is building a development environment for the Xilinx FPGA devices that can be used by any application developer.
 - Included in this environment is the software equivalent of a device driver for the FPGAs.
 - Each interface in the FPGA is fully defined in a Hardware Description Language (HDL) module.
 - Essentially these modules wrap around the application's HDL and provide a simple driver to each external interface of the device.
 - An Interface Description Document (IDD) is being written to help the applications developer communicate with these modules and a system simulation environment is available so the application developer can fully simulate their application prior to integration into the payload.

- Reconfigurable systems technology has been successfully applied to the terrestrial domain.
- Significant issues must be addressed to successfully deploy the same capability in space, e.g.
 - Size, Weight and Power (SWAP) constraints
 - Radiation effects
 - Long-term deployment
 - Remote access

- One area of concern when integrating an application into a space vehicle is proving that the application will not harm the hardware or other payloads on the spacecraft.
- In developing these "module wrappers", fail-safe mechanisms, like range checking, are employed to ensure that faulty data cannot propagate to the external interfaces.
- Once these interfaces are qualified against the do-not-harm requirement, the application developer can use them without repeating qualification testing - that will lead to a faster buyoff for flight operations.

Standards Based Element Connection



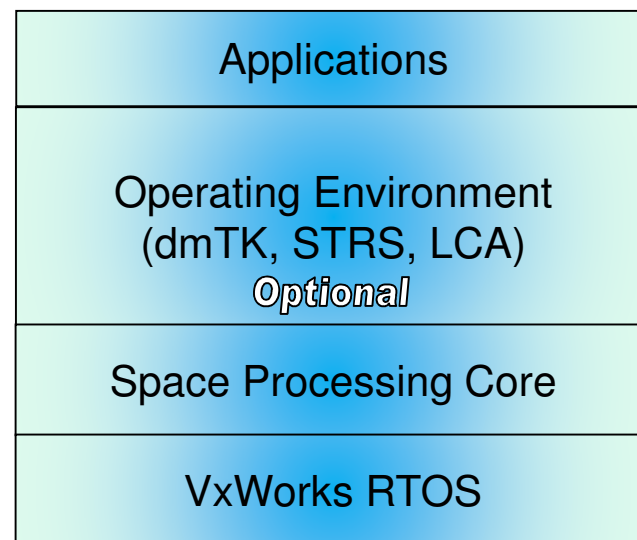
- We chose to implement an open architecture, using standards as the basis to connect the elements together.
 - In doing so, a processing element can be created from any processing technology.
 - As long as the processing element is compliant with the processor module's Internal Connection Document (ICD), the module can be integrated into the system.
- To the GPP Subsystem, the module looks identical to all other modules in the system – the only difference will be the type of executable that is loaded into its memory.
- Memory maps will be different between modules, but only because of the amount of memory available.
 - Similar to the Control Status Registers (CSRs) that exist in cPCI, a standard set of control registers will be allocated to a common set of memory addresses, so the GPP can distinguish the type of module and make the appropriate interface decisions.

- Multiple standards have evolved for the management of software reconfigurable radio systems:
 - The *Software Communications Architecture* (SCA) developed in conjunction with the Joint Tactical Radio System (JTRS) program.
 - The *PIM and PSM for Software Radio Components* developed by the Object Management Group (OMG)
 - The *Space Telecommunications Radio System* (STRS) developed by NASA.
- Several NASA programs use the STRS specification for the control interfaces to the radio system.

Software Operating Environment



- The Space Processing Core is legacy software that provides the necessary tools needed to operate an on-board processor
 - Application tasking control via a script interpreter
 - Abstraction layer for application access to communication interfaces
 - Command and Telemetry processing
 - Provides mechanisms for Fault management
 - Flying on PCIP (> 8 years of flight time)
- The RTOS provides the main interface between the hardware and the space processing core
 - Initial systems has used VxWorks
 - We have traded other COTS systems, specially to resolve high mission assurance requirements
 - ARINC 653 has been evaluated for it applicability
- The Operating environment manages the communications and signal processing applications and components.
 - To provide 3rd party programmability, an SCA-like OE is selected for its maturity and ability to port new applications

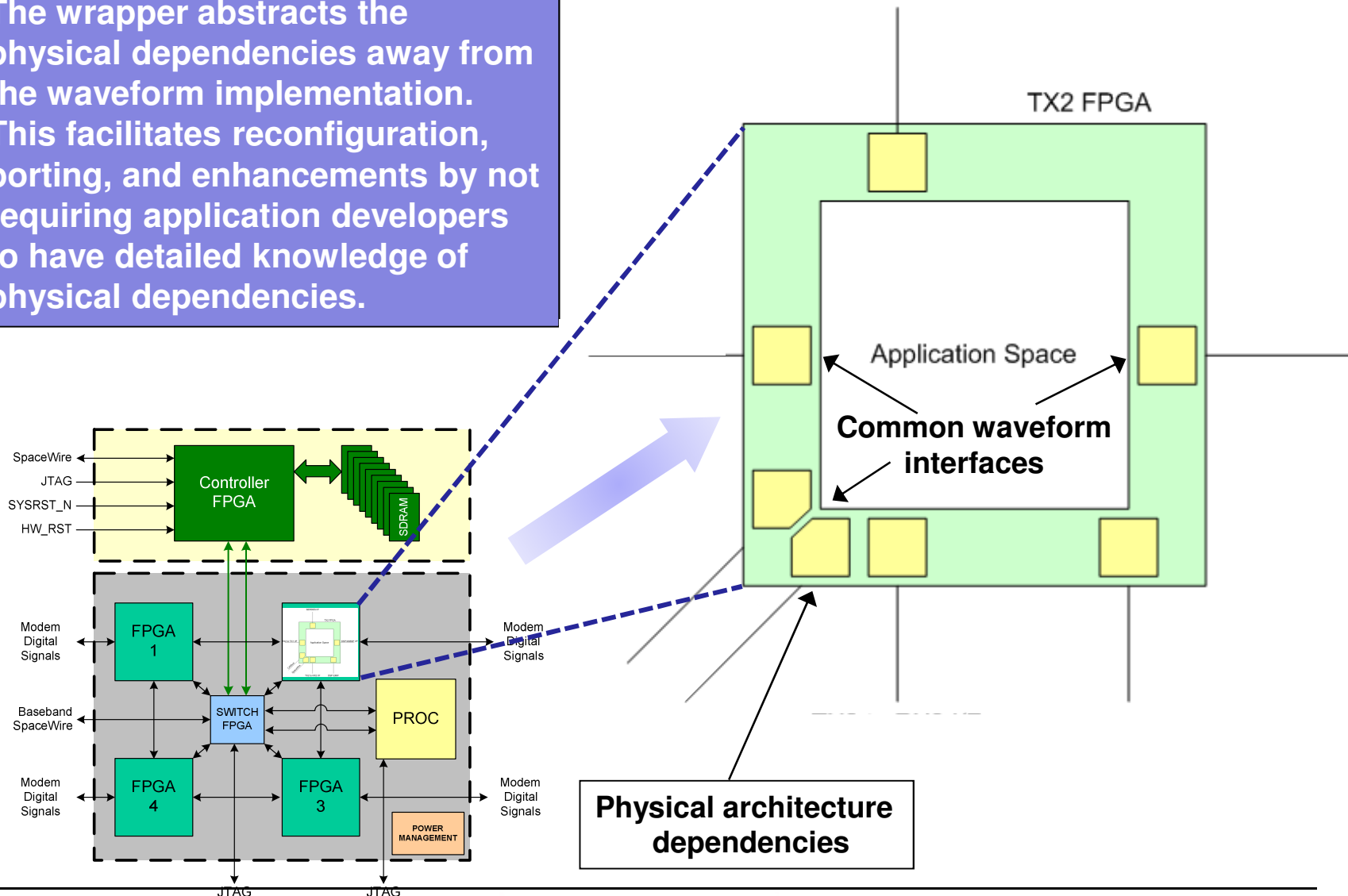


- A toolkit is being created to assist programmers in developing FPGA applications
- In this system, the application developer is given a configurable code set where each interface can be selectively included
 - Removing unused interfaces increases overall gate capacity
- Each interface is designed for space mitigation and will be certified to cause no harm to the overall system
 - This certification is handled through range and protocol checking
 - An ICD will be delivered with toolset to define the application interface
- Multi-FPGA simulation models will be provided that help validate the user's application in a simulated modem environment
 - Includes all commanding interfaces
- Demonstrated Module Wrapper in April 2008
 - Ported an existing OFDM transmit waveform operating at 54 Mbps to the modem
 - Porting of this 3rd party waveform took less than two weeks.

Module Wrapper Concept



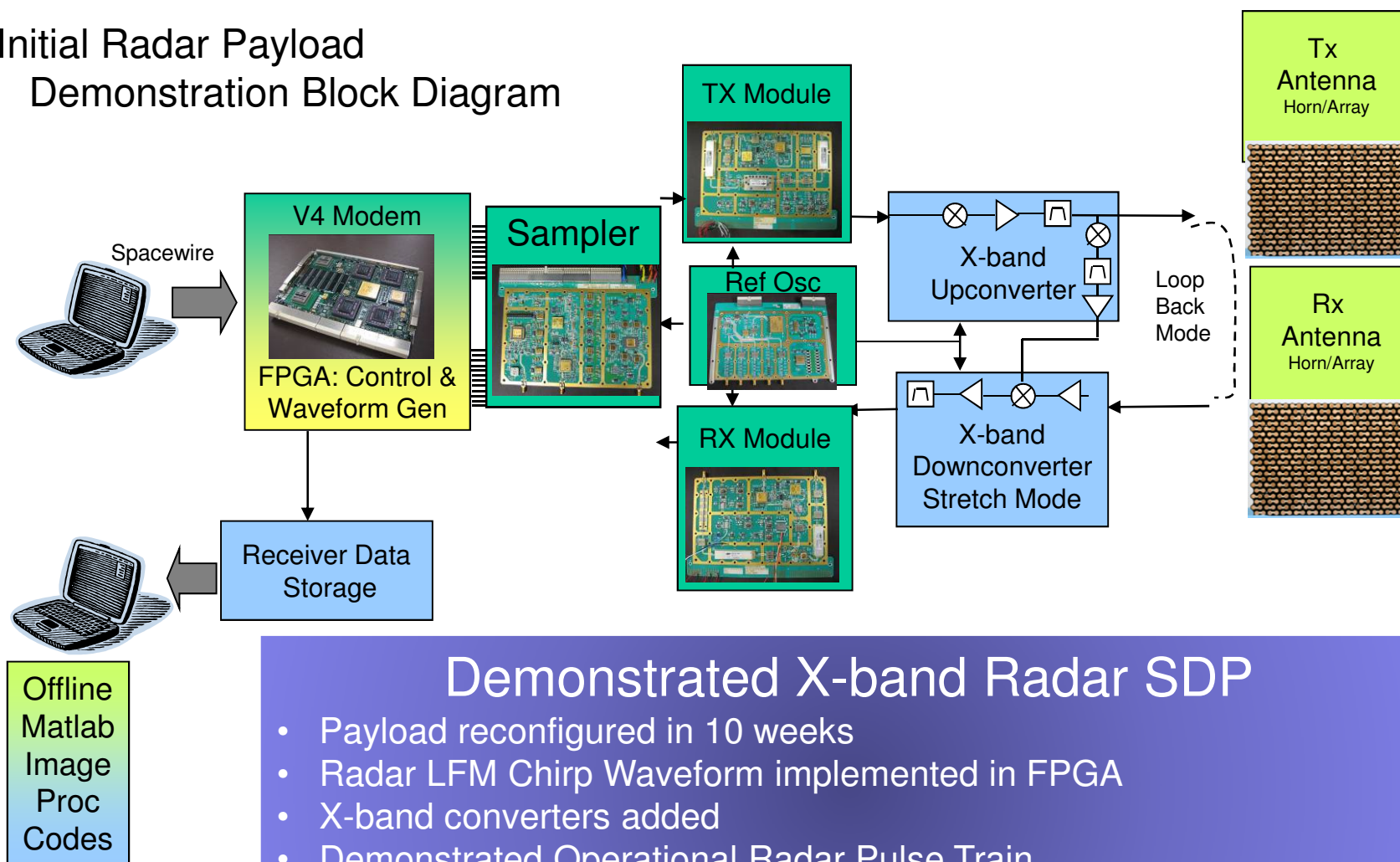
- The wrapper abstracts the physical dependencies away from the waveform implementation.
- This facilitates reconfiguration, porting, and enhancements by not requiring application developers to have detailed knowledge of physical dependencies.



SDP Reconfigured for Radar Payload Demonstration– June 2010



Initial Radar Payload
Demonstration Block Diagram



Demonstrated X-band Radar SDP

- Payload reconfigured in 10 weeks
- Radar LFM Chirp Waveform implemented in FPGA
- X-band converters added
- Demonstrated Operational Radar Pulse Train
- Measured IPR/MNR Plots

SDP Capabilities Enable 3rd Party Waveform Porting



- Operational SDP COM & RADAR Payloads demonstrated
- Software Defined Payload Infrastructure
 - Low cost & rapid demo/risk reduction for small payloads
 - Reconfigurable to support mission flexibility
- COM Software Defined Radio Expertise
 - Harris supports five different SDR platforms – Waveform Library
 - Experience porting waveforms between platforms
- Flight Qualified TRL-7 Software Defined Payload
 - Reconfigurability reduces program execution risk and development time
- Mission Payload Algorithm Expertise
 - COMM & ISR
- 3rd Party Software Compatibility through STRS & SCA standards

*SDR, Radar, & DSP
Waveform Library*



Software Defined Payload Infrastructure Enables Low Cost, Reconfigurable Mission Performance with Rapid Demonstration Capabilities

- Careful planning of Spaceborne SDR architectures can enable 3rd party waveform porting through the use of
 - A development environment suitable for any developer including
 - Standard interface descriptions and definitions
 - Module wrappers for application's an HDL providing a simple driver for each external device interface
 - Spaceborne specific requirements such as SWaP, radiation effects, long-term deployment and remote access
- Continuation for future scalable platforms uniformly constructed to include payloads ranging from small (ORS for example) to large (TSAT-like) platforms
- Software Defined Processing for multi-mission applications