Universal Reconfigurable Translator Module (URTM)

Military and Aerospace Programmable Logic Devices (MAPLD) Conference 2009
August 31 – September 2, 2009

Edward Leventhal
Roman Machan
Rob Jones

Excellence in Aerospace Technology

www.sigmaspace.com

Abstract

Sigma Space Corporation, under contract to NASA Langley Research Center (LaRC), has developed a Universal Reconfigurable Translation Module, or URTM, in order to translate, both logically and physically, specific serial protocols of interest to NASA. The initial prototype configuration targeted MIL-STD-1553B (both RT and BC), ECSS-E-50-12A (SpaceWire), and IEEE 1394b (FireWire). The URTM is FPGA-based, making use of both Actel and Xilinx devices. The URTM is comprised of two basic circuit card types: a Reconfigurable Protocol Translator Module (RPTM), which is responsible for the actual protocol translation and a Protocol Interface Module (PIM), which provides the physical interface for the given protocol. By simply connecting two of the three PIMs on either end of the RPTM, the URTM then self configures via a library of interface translation functions, thereby allowing the two data links to communicate seamlessly. The URTM is reconfigurable and expandable, since the same hardware can be used to perform translations between different protocols by adding a simple new hardware PIM for the physical interface and new VHDL-based logic for the particular link layer.

The URTM has a small size (3” x 3.44” x 1.40”) and low power (approximately 6 W, depending on the PIM combination), and a clear migration path from the developed engineering model to a spaceflight unit. Key features for spaceflight include the use of components with flight-equivalent parts, interconnections to allow the radiation tolerant Actel command/control FPGA to scrub the SRAM-based Xilinx translation FPGA, resource utilization to allow for TMR of the Xilinx FPGA, triplicated I/O (where possible) at the PCB level for the Xilinx FPGA, conductively cooled mechanical design, double-step corners for EMI, outgassing holes, flying leads on external connectors for mechanical decoupling, and a robust interconnect suitable for high vibration environments.

The protocol translation engine was initially modeled in Python. The structure of this model was maintained as the URTM hardware design progressed. The modular architecture of the URTM is designed to separate the physical interfaces from the link layer protocols in order to facilitate the addition of future communication protocols. In addition, the VHDL partitioning implements well-defined blocks for reading, writing, and translation functions that allow for much re-use if adding a new protocol. These features make the URTM hardware more “universal” than a translation of just one protocol to another.

Bit rates of 1 Mbit/sec for 1553B, 200 Mbit/sec (outbound) and 120 Mbit/sec (inbound) for SpaceWire, and 800 Mbit/sec for 1394b were attained. The protocol translation was successfully modeled and was demonstrated in hardware to NASA at these bit rates using all valid PIM combinations.

The URTM may serve a role where a flight-proven legacy piece of equipment is to be used in a project using newer interfaces (such as a piece of equipment with a 1553B interface that must be controlled by a SpaceWire or 1394b-based computer). In such a case, the URTM can serve as the bidirectional translator thereby saving hardware redesign cost and time.

Outline

- Overview and Objectives
- Design and Development
- Future Work

Technical Objectives

- Model, design, and develop a radiation-hardened, universal serial data link translator

- Support popular data links for space applications:
  - ECSS-E-50-12A (SpaceWire) @ 125 Mbit/sec (Goal: 200 Mbps)
  - MIL-STD-1553B (BC & RT) @ 1 Mbit/sec
  - IEEE 1394b (FireWire) @ 400 Mbit/sec (Goal: 800 Mbps)

- Adapt any two point-to-point data links logically, electrically, and mechanically

- Automatic reconfiguration
  - Connect the Physical Interface Modules (PIMs) on either end of the adapter
  - The Reconfigurable Protocol Translation Module (RPTM) self-configures via a library of cores
  - The two data links communicate seamlessly

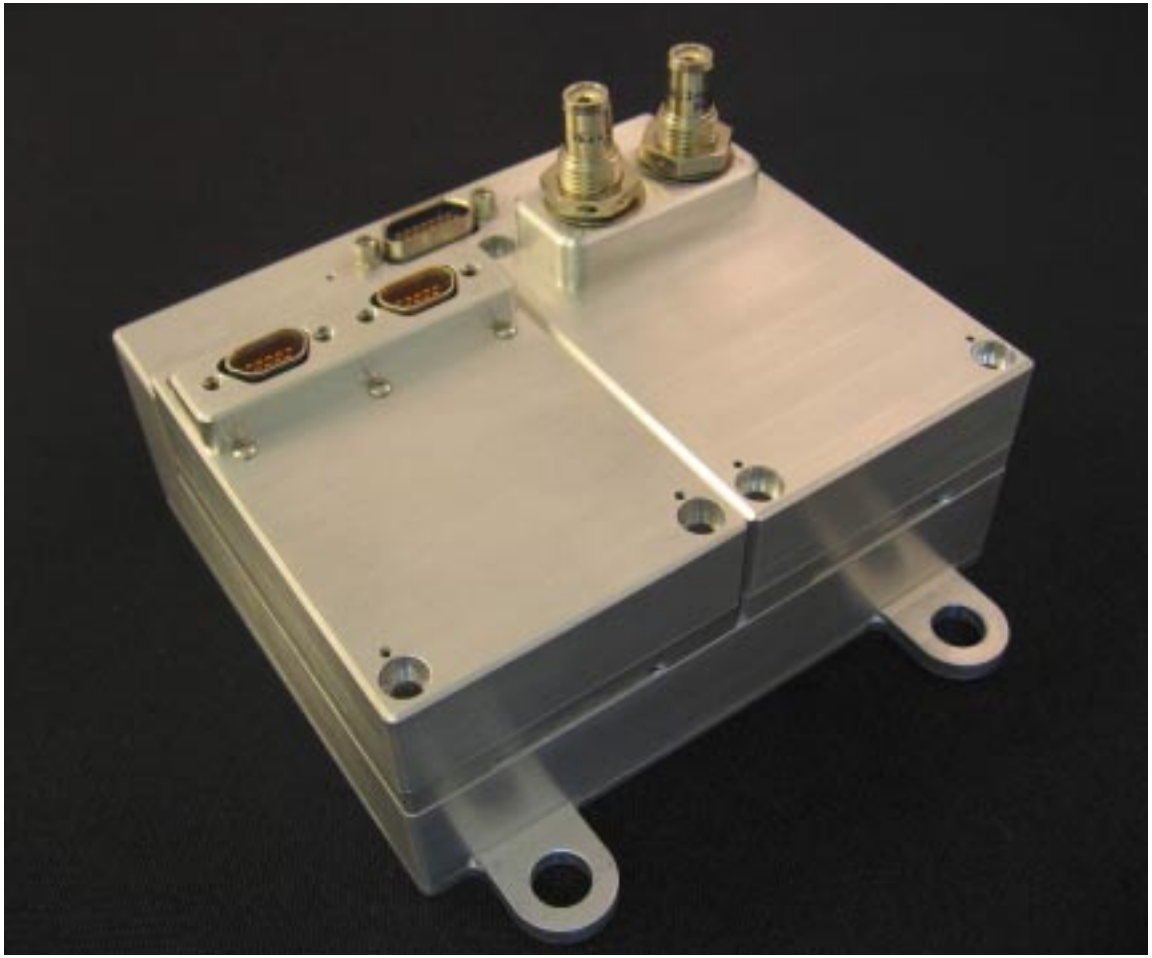
Design for Space-Flight

Unit built for proof-of-concept validation has been designed for space-flight applications:

- Hooks for scrubbing (Actel / Xilinx) via Slave SelectMap Interface
- Use of flight-like or flight-equivalent components
- Resources in Xilinx for TMR and other SEE mitigation approaches
- Triplicated I/O on Xilinx except for bi-directional signals
- Conductively cooled mechanical design
- Double-step corners for EMI
- Outgassing holes
- Flying leads on external connectors for mechanical decoupling
- Robust interconnect with ample compliance in the “Z” direction for high vibration environments

URTM

Packaged URTM:
RPTM, SpaceWire PIM, and 1553B PIM



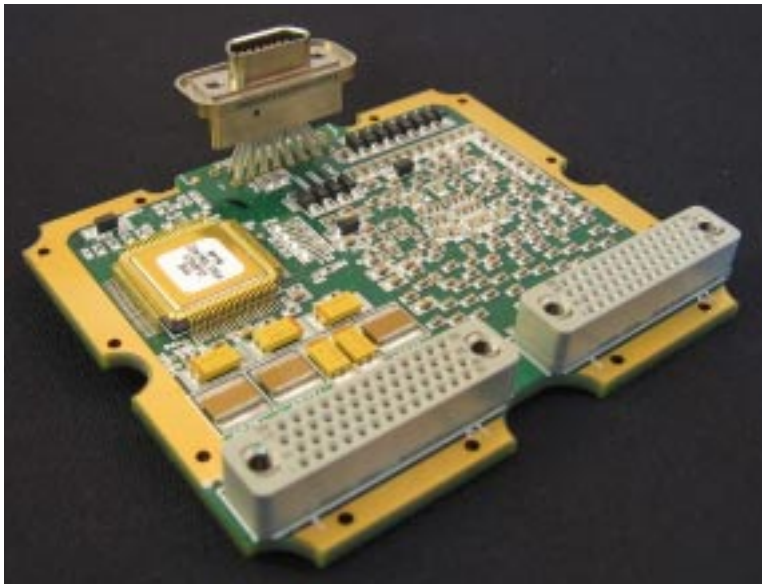
Volume, Mass, Power

Module	Mass (grams)	Mass (oz)	Power (Watts)
RPTM	220	7.8	5.61
1553b PIM	75	2.6	0.132
SpaceWire PIM	65	2.4	0.337
1394 PIM	80	3.0	0.825

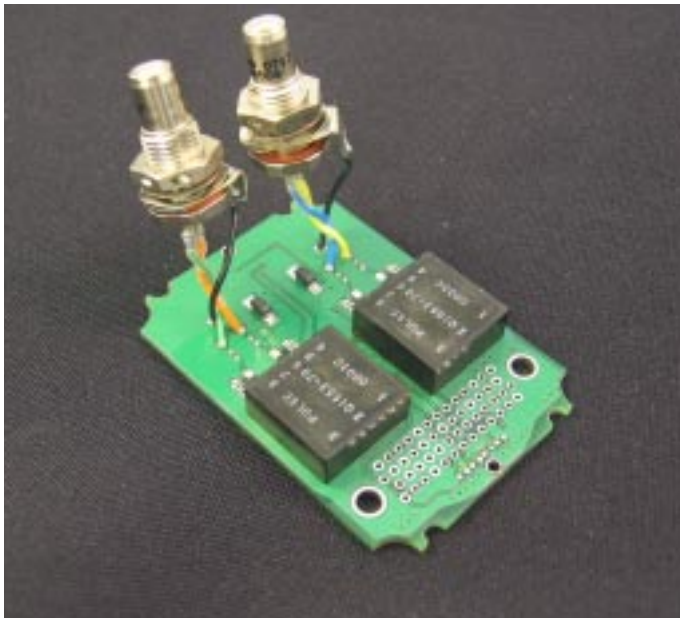
Total Volume for the URTM
with SpaceWire and 1553 PIMs
(excluding projections for connectors)
is approximately:

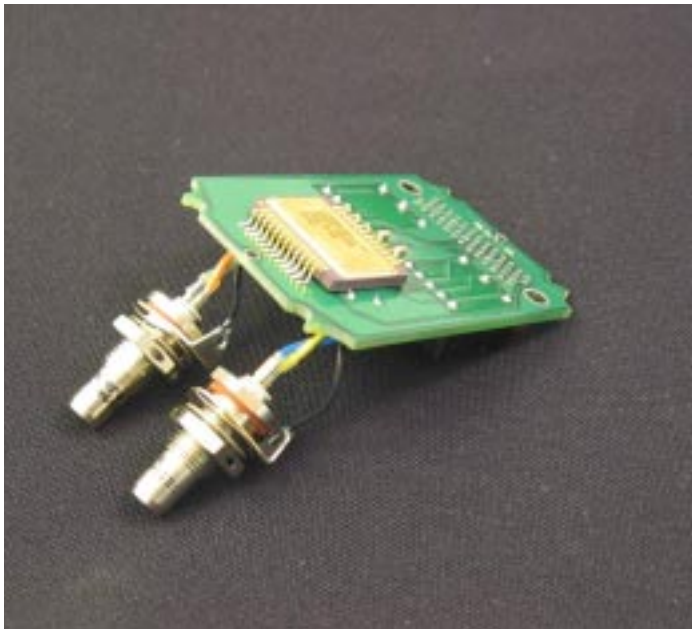
3” x 3.44” x 1.40”
(76mm x 87.5mm x 37mm)

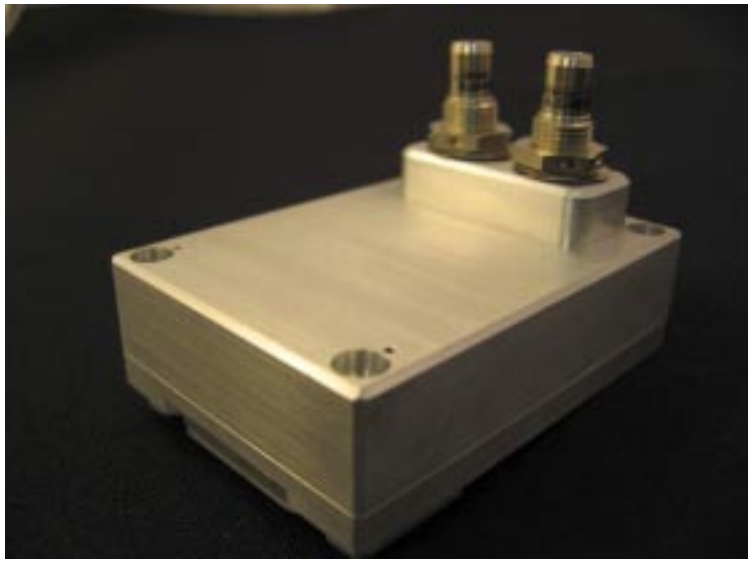
RPTM



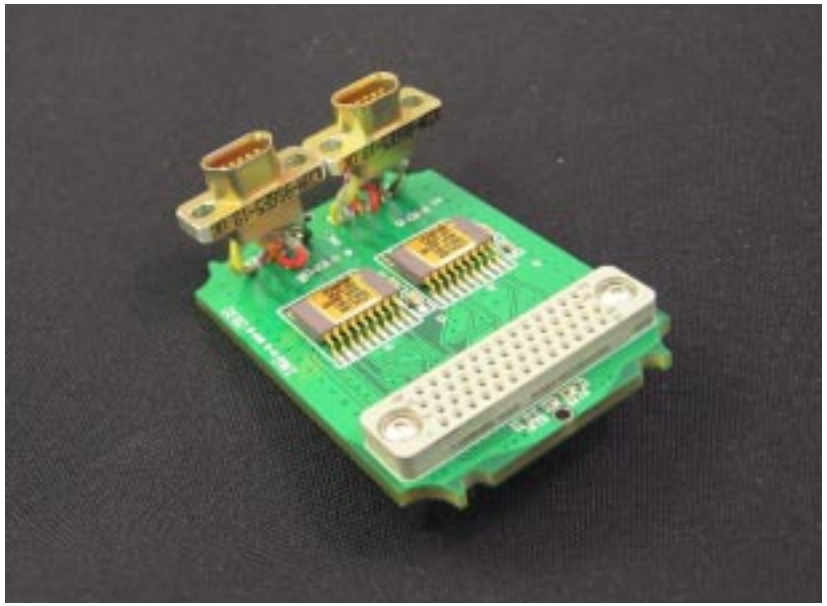
1553B Physical I/F Module



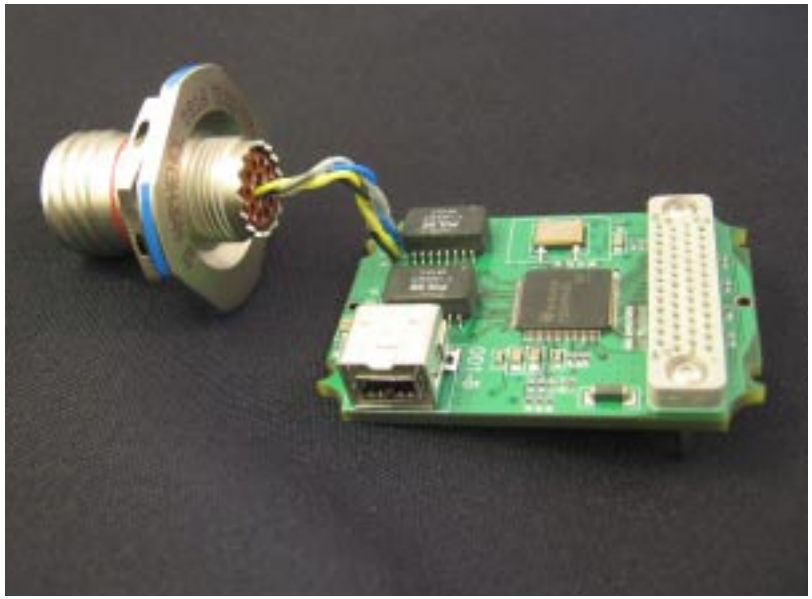


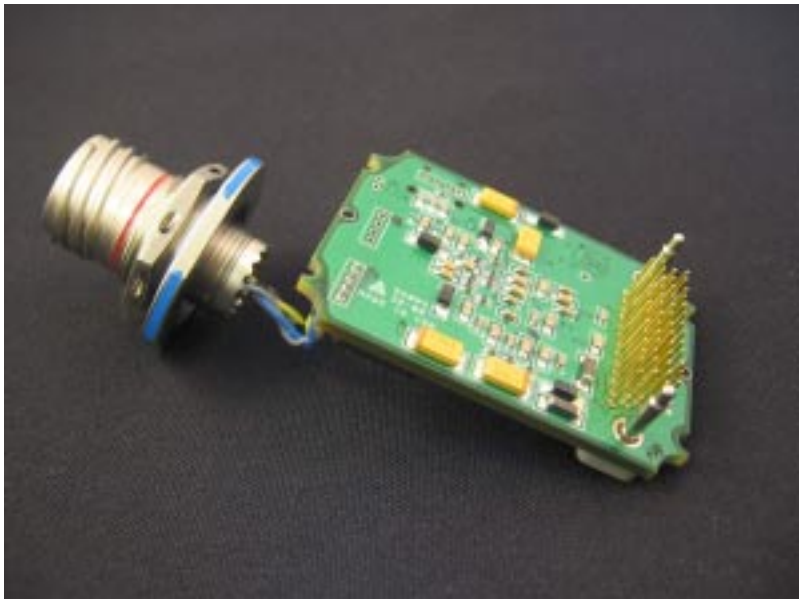


SpaceWire Physical I/F Module

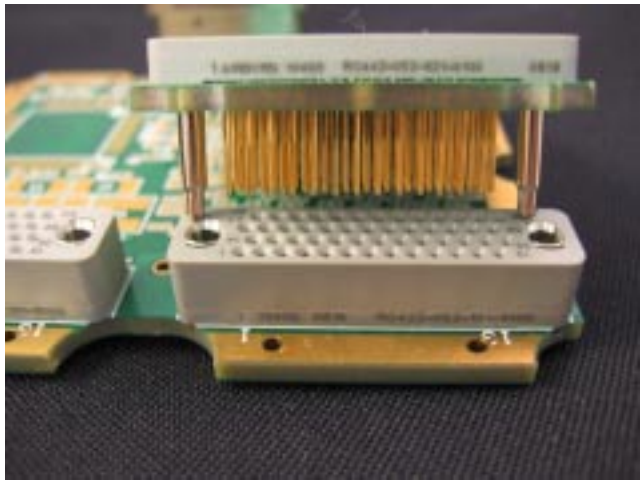


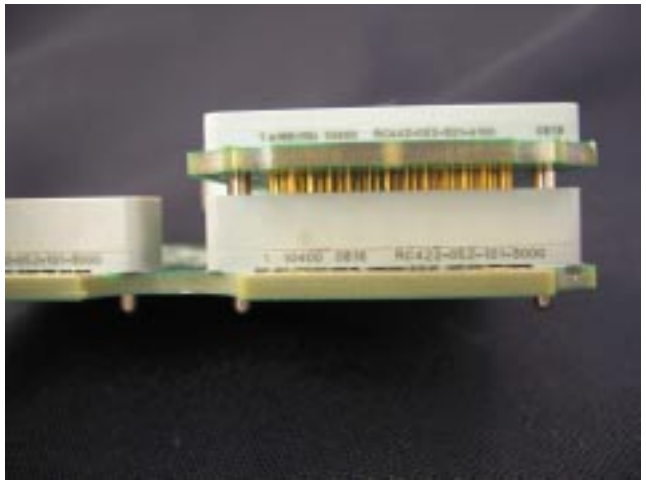
1394b Physical I/F Module



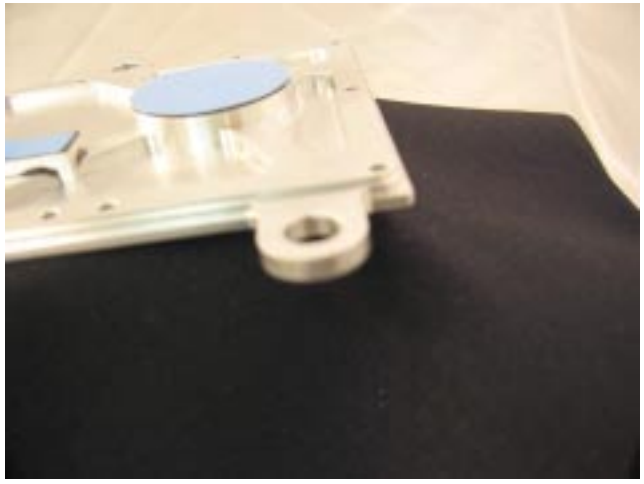


URTM / PIM Fabrication





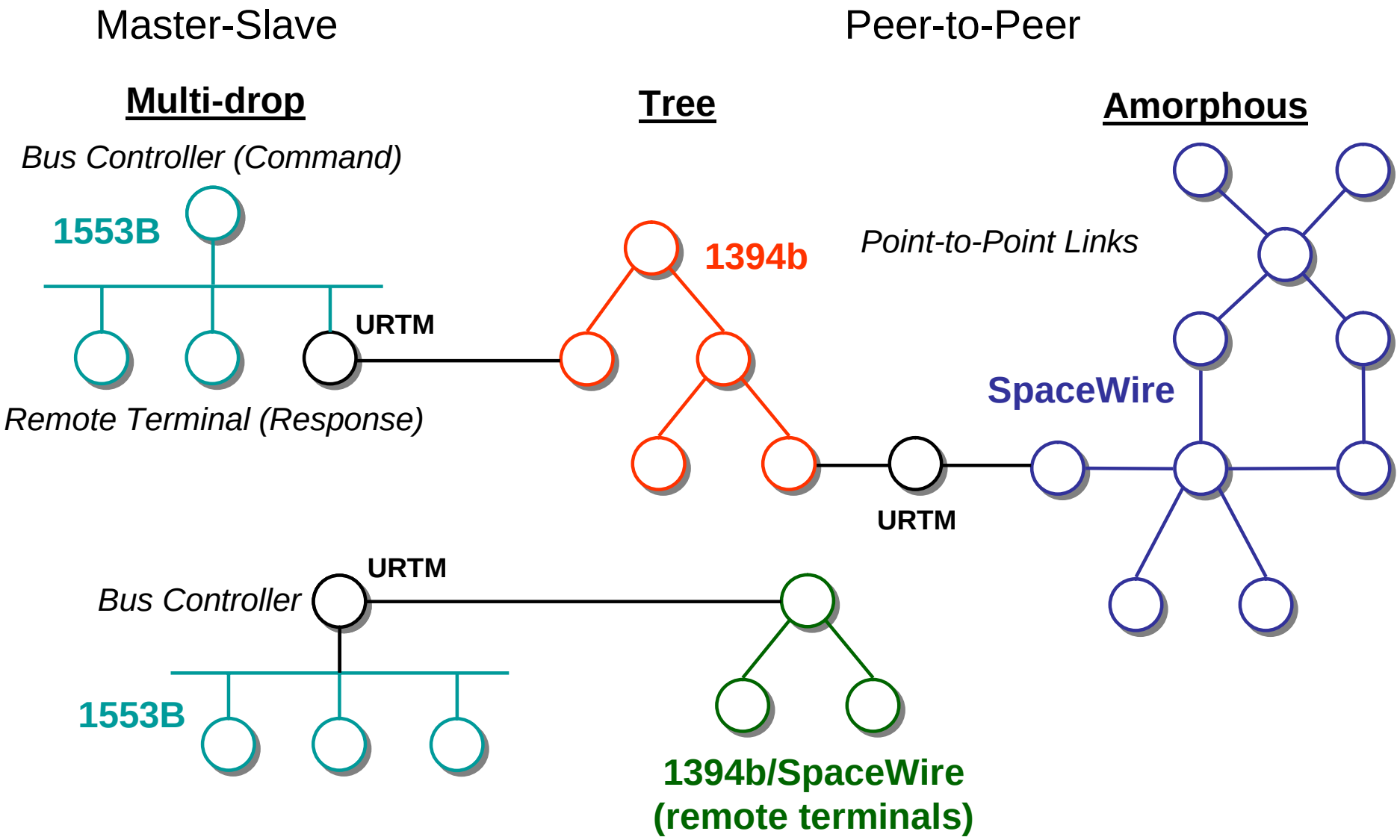
Robust interconnect with ample compliance in the “Z” direction for high vibration environments.



EMI enhanced enclosure with double corner design

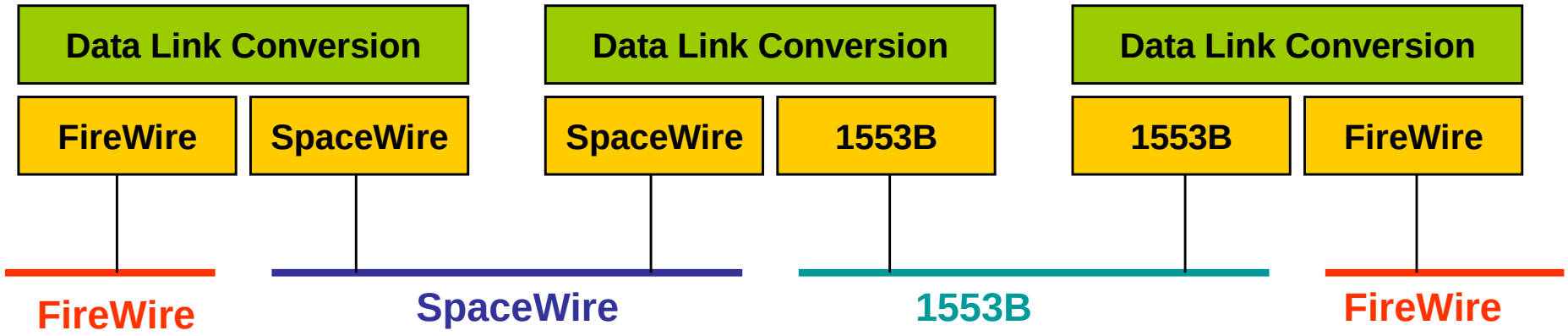
External Connectors are mechanically decoupled via “Flying Lead” connections for all PIM I/O as well as for the console port connector.

URTM is a Network Bridge

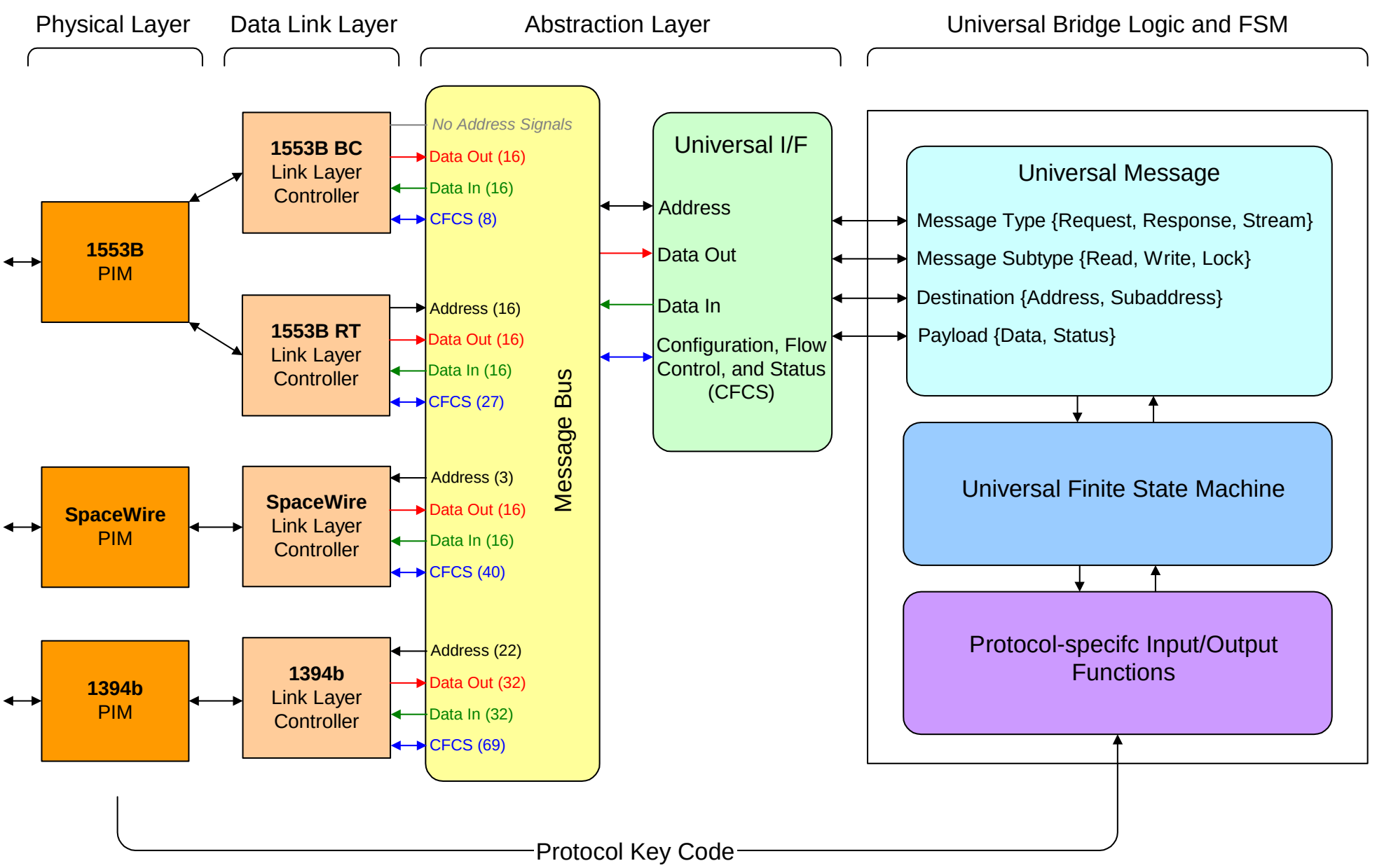


Protocol Overview

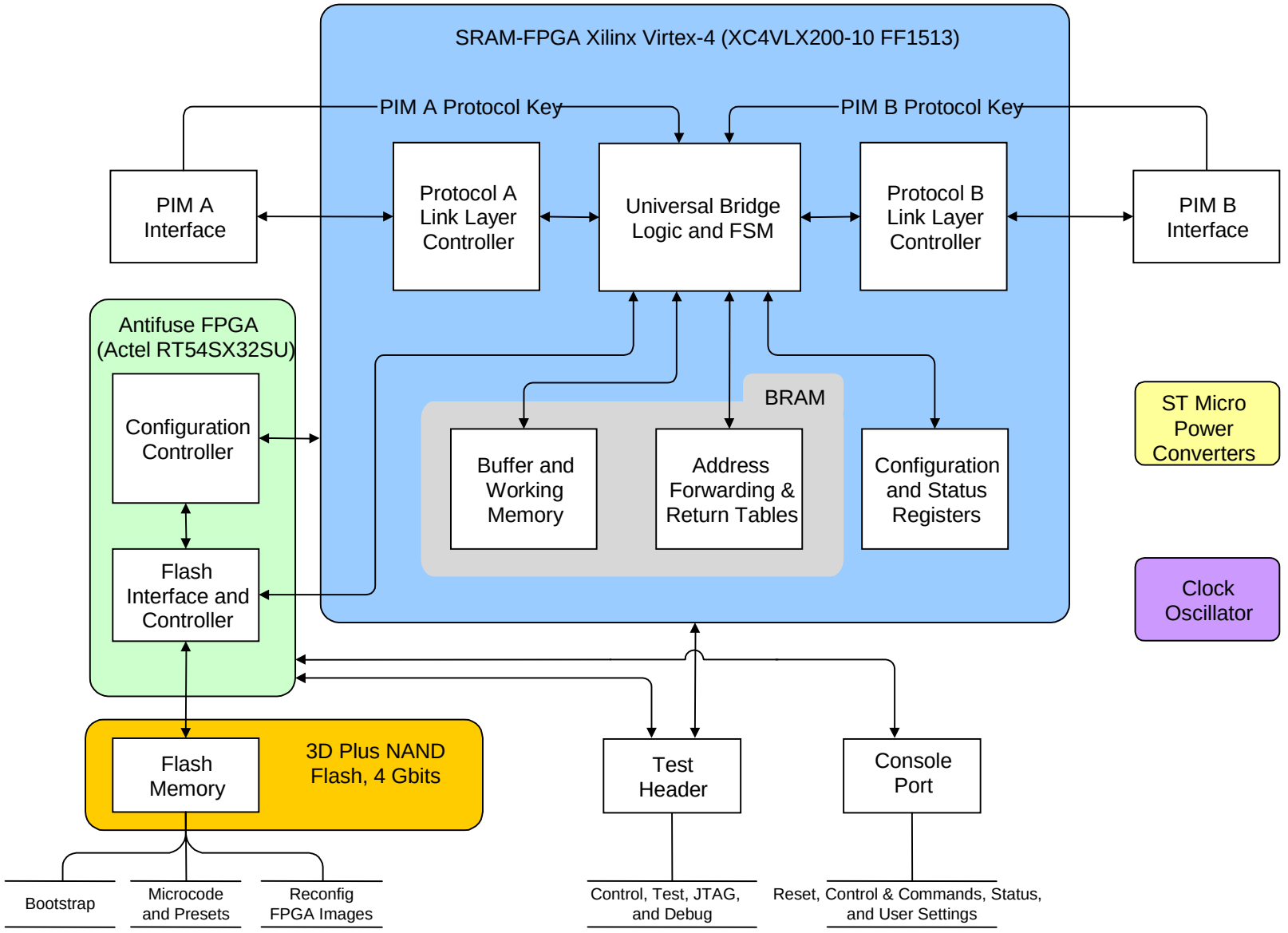
Node ID	**8 bits**	**5 bits**	**16 bits**
Subaddress Space	**undefined**	**5 bits**	**48 bits**
Transfer Unit	**10 bits (8-bit data)**	**20 bits (16-bit data)**	**Packet (32-bit data)**
Max Packet Length	**infinite**	**32 data words**	**Rate Dependent**
Data Rate	**2-400 Mbps**	**1 Mbps**	**100,200,400,800 Mbps**
Data Link	**Point-to-Point**	**Multi-drop**	**Point-to-Point**
Networking	**Peer-to-Peer**	**Master-Slave**	**Peer-to-Peer**
Operation	**Full-duplex**	**Half-duplex**	**Half/Full-duplex**



Architecture Layers



RPTM Block Diagram



FPGA Functions

Actel FPGA

- Provides a console (UART) interface for user command and status from both the Actel & Xilinx FPGAs
- Provides a command/status interface to the Xilinx in order to forward received commands and to accept status from the Xilinx FPGA
- Provides a slave SelectMap interface to the Xilinx FPGA in order to perform configuration (and in the future, readback and scrubbing, if desired).
- Provides a controller for the Flash memory stack that can identify bad blocks, perform page reads and writes, perform block erases, and interface with the SelectMap controller block.

Xilinx FPGA

Provides all buffering functions, reader, interpreter, writer, Forward Address Table (FAT), and Return Address Table (RAT) functionality for the translation process.

Key IP Cores & Components

RPTM Xilinx On-chip IP Cores:

- 1553 RT: Sital RT1553FE
- 1553 BC: Sital BC1553FE
- SpaceWire: NASA GSFC Space Wire Core (CD Date: 09/12/2007)
- 1394b: DAP Technologies FireLink Basic

Key RPTM Devices & Interfaces:

- Oscillator: 125 MHz
- Configuration Memory: 4Gbit 3D-Plus stacked NAND FLASH
- Configuration Control: Actel A54SX32A-CQ84 (RT54SX32S-84CQFP)
- Translation: Xilinx XC4VLX200-10 FF1513 (XQR4VLX200-10 CF1509)
- Test Header: Xilinx JTAG, mode pins, and assorted test points
- Test Header: Actel JTAG and probe signals
- Console Port: RS-232 protocol, 8N1, 115200 bps

Key PIM On-Card Devices & Interfaces:

- 1553: Aeroflex transceivers with Pulse transformers
- SpaceWire: Aeroflex LVDS drivers / receivers
- 1394b: TI 1394B 3-Port cable transceiver / arbiter with transformers

FPGA Utilization

Xilinx FPGA

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	6,031	178,176	3%	
DCM autocalibration logic	42	6,031	1%	
Number of 4 input LUTs	10,365	178,176	5%	
DCM autocalibration logic	24	10,365	1%	
Logic Distribution				
Number of occupied Slices	9,065	89,088	10%	
Number of Slices containing only related logic	9,065	9,065	100%	
Number of Slices containing unrelated logic	0	9,065	0%	
Total Number of 4 input LUTs	10,996	178,176	6%	
Number used as logic	9,568			
Number used as a route-thru	631			
Number used as 16x1 RAMs	56			
Number used for Dual Port RAMs	640			
Number used as Shift registers	101			
Number of bonded IOBs				
Number of bonded	282	960	29%	
IOB Flip Flops	15			
Number of BUFG/BUFGCTRLs	13	32	40%	
Number used as BUFGs	12			
Number used as BUFGCTRLs	1			
Number of FIFO16/RAMB16s	76	336	22%	
Number used as RAMB16s	76			
Number of DCM_ADVs	6	12	50%	

Actel FPGA

SEQUENTIAL	Used: 742 Total: 1080 (68.70%)
COMB	Used: 1432 Total: 1800 (79.56%)
LOGIC *(seq+comb)*	Used: 2174 Total: 2880 (75.49%)
IO w/ Clocks	Used: 52 Total: 59
CLOCK	Used: 2 Total: 2
HCLOCK	Used: 1 Total: 1

Resources are available for implementation of scrubbing function.

Utilization for Xilinx FPGA represents instantiations of all IP cores as well as control logic.
Resources are available for TMR implementation.

Modeling

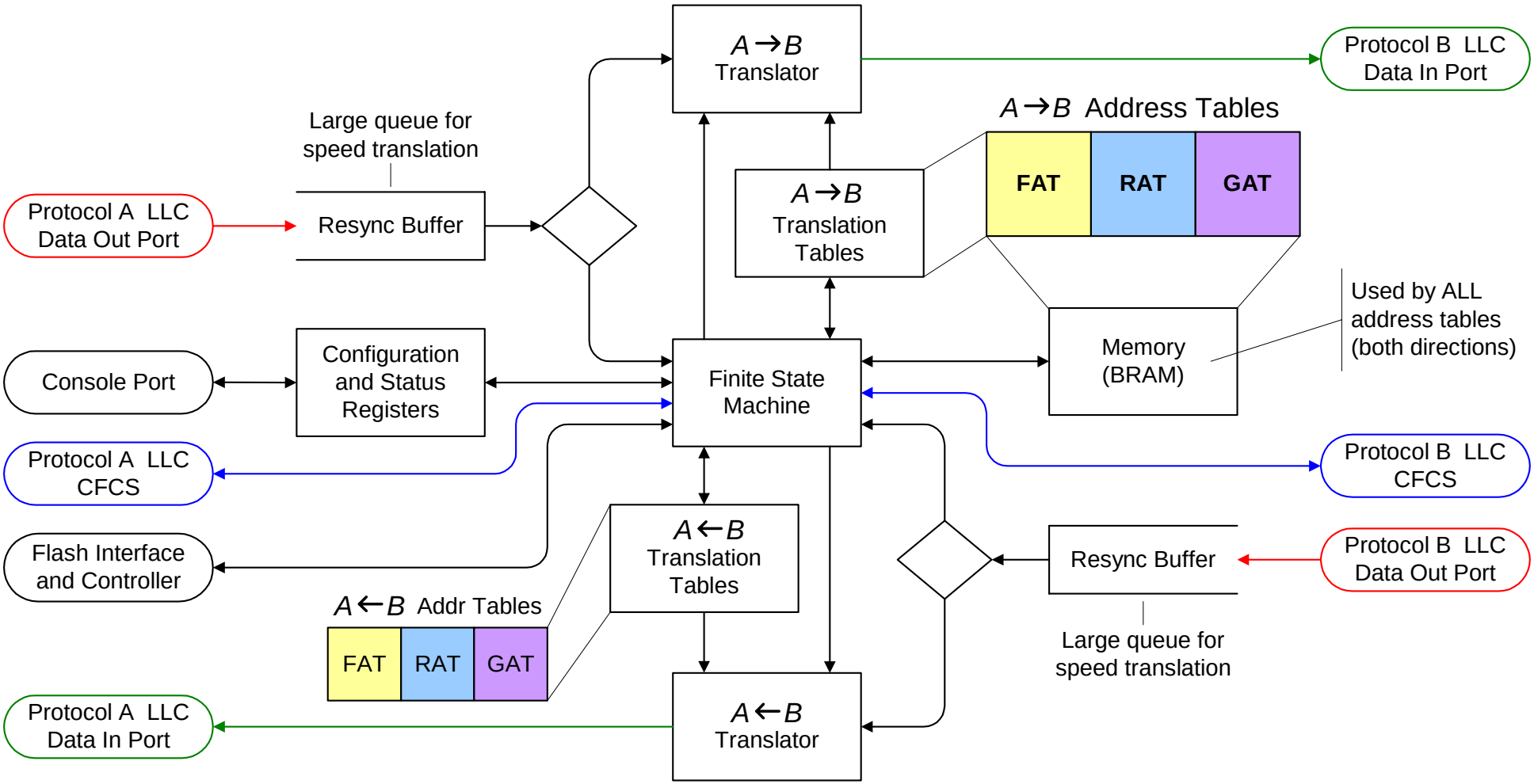
- Python-language development models were created to capture the packet transactions, data flow, and control flow.

- These models were useful when conducting concept and feasibility studies prior to, and during design and implementation.

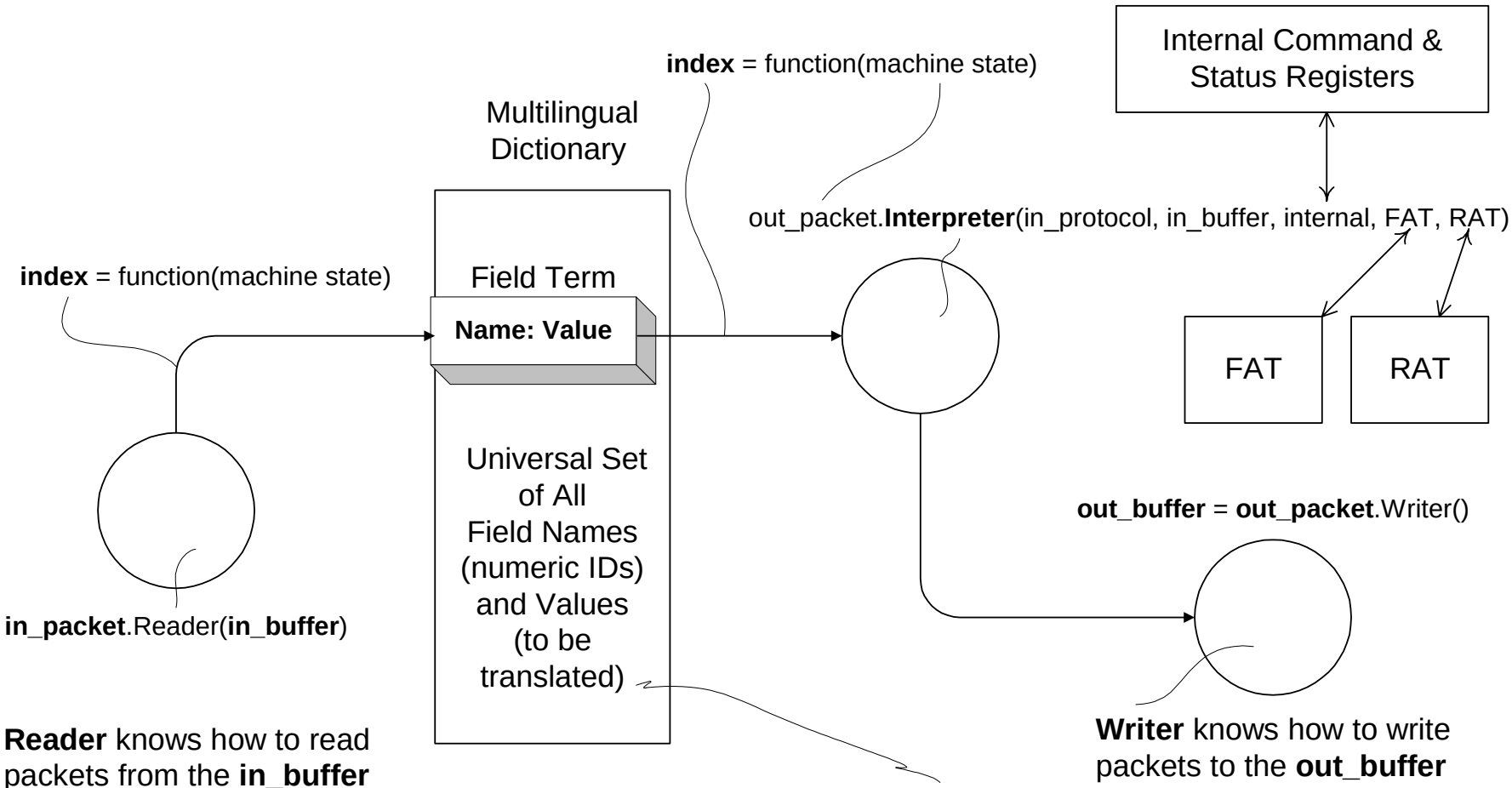
- Python models also assisted the logical design and verification, subsequent performance evaluations, and provided a basis for the FPGA-targeted translator architecture and design.

- Independent validation of functionality via higher-level modeling has proven useful and has reduced the number of functional, logical, and control errors that would otherwise have dominated the design and debug effort.

Dataflow Model



Bilingual Dictionary Model



Reader knows how to read packets from the **in_buffer**

Readers are state machines, each customized to a particular **in_packet** protocol

The data structure for this **dictionary** can be implemented with a hash table

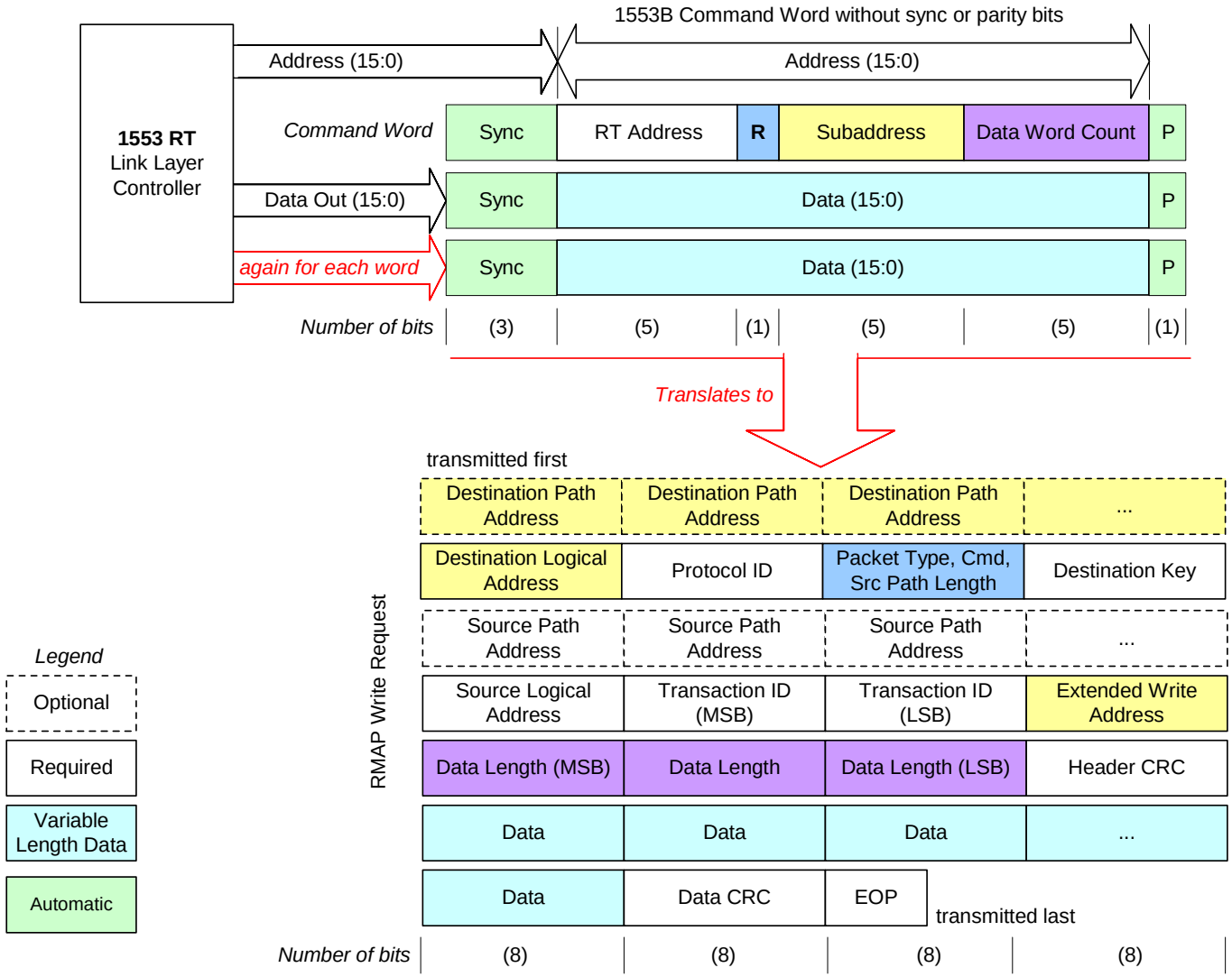
The table **indices** are functions of the states that sequence through the packet fields in correct order

Writer knows how to write packets to the **out_buffer**

Writer are state machines, each customized to a particular **out_packet** protocol

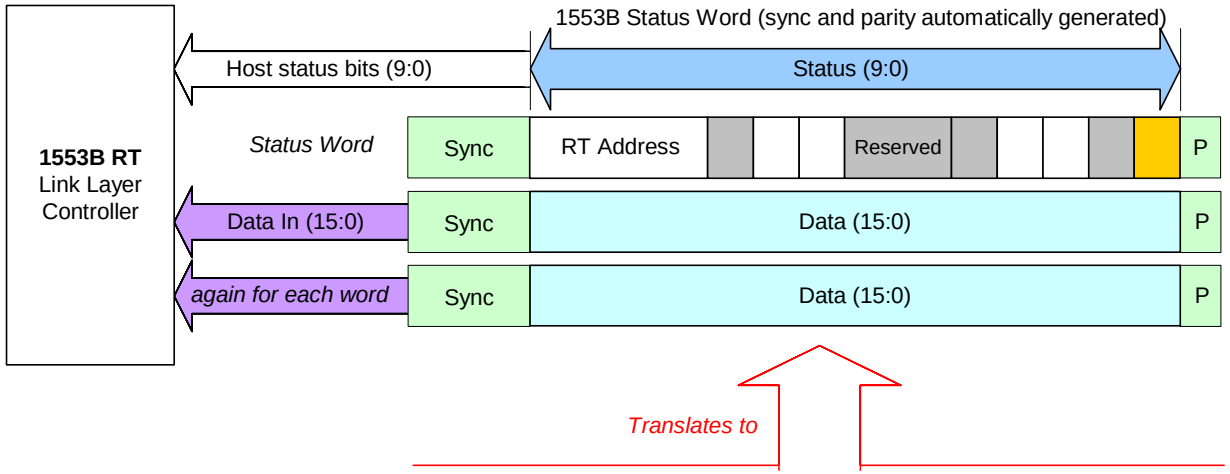
Translation Mapping - Example

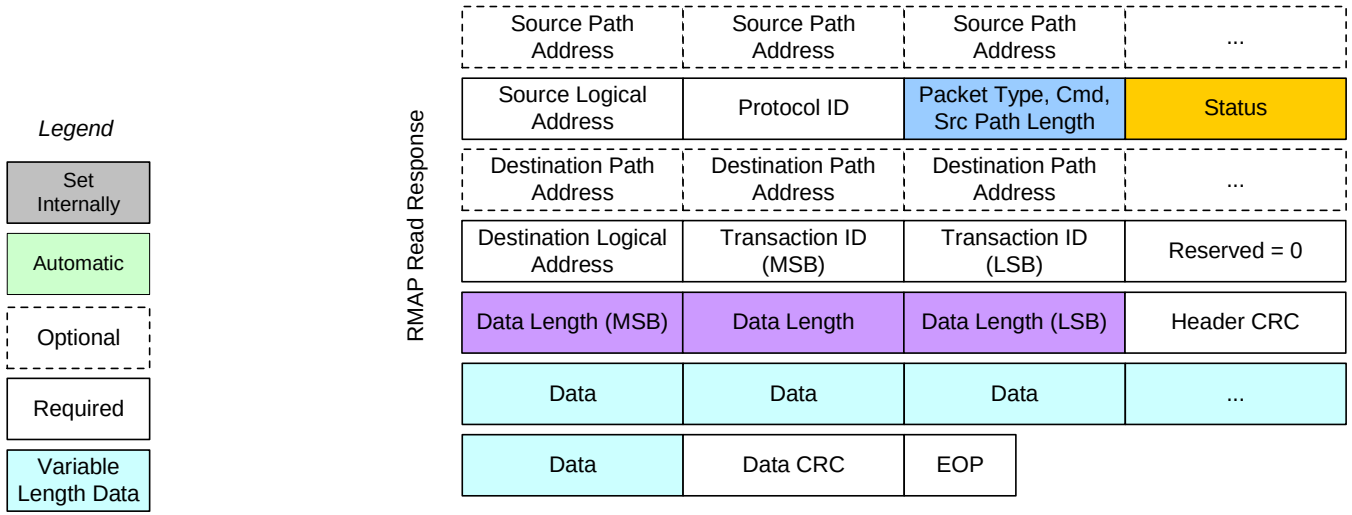
1553-to-SpW RMAP Transaction: RT Receive Translates to SpW RMAP Write



Translation Mapping - Example

1553-to-SpW RMAP Transaction: Read Response Translates to RT Transmit Response

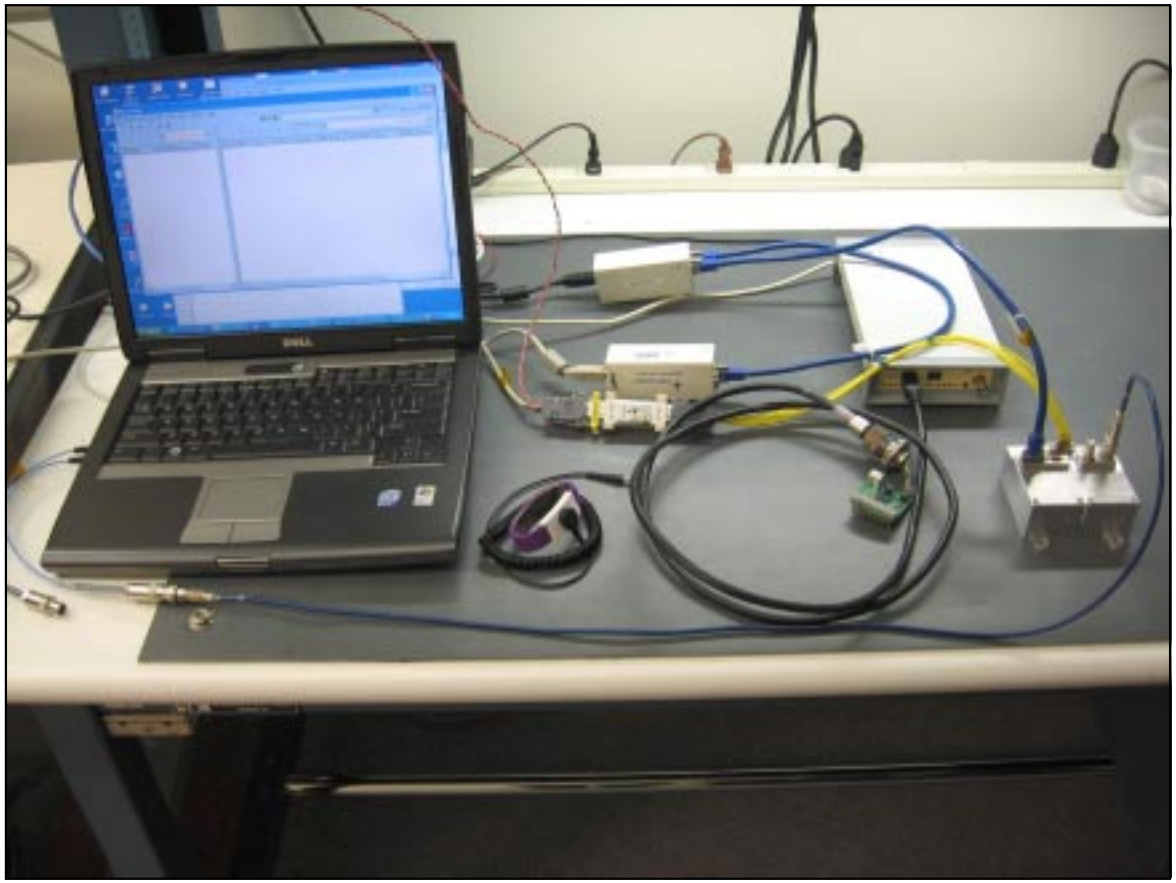




URTM Testing

Independent COTS Testers Validated Protocol Translation At Desired Rates

SpaceWire:	STAR-Dundee USB Brick & Link Analyser	-	200 Mbits/sec Tx, 120 Mbits/sec Rx
1553B:	Excalibur Systems PCMCIA Tester	-	1 Mbit / sec
1394b:	DAP FireSpy	-	800 Mbits/sec

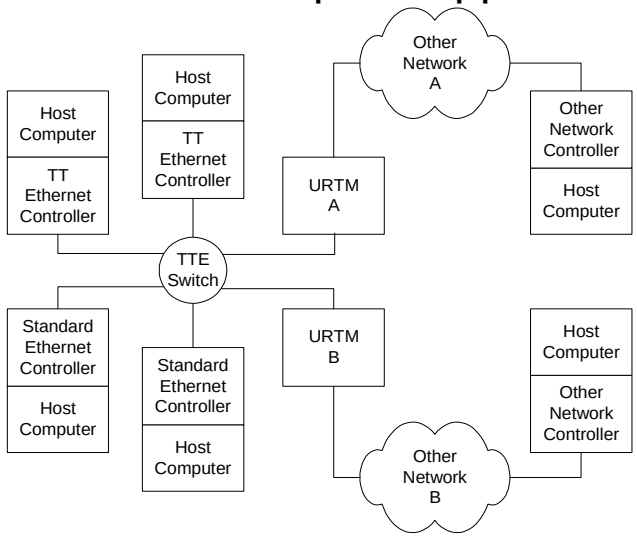


Transactions Demonstrated

Tester	Core	Core	Tester
RT, Sends Write Response	1553BC	SpW	Receives Write Response
RT, Sends Read Response	1553BC	SpW	Receives Read Response
RT, Sends Write Response	1553BC	1394b	Receives Write Response
RT, Sends Read Response	1553BC	1394b	Receives Read Response
BC, Sends Write Command	1553RT	SpW	Receives Write Command
BC, Sends Write Command	1553RT	1394b	Receives Write Command
SpW Sends Write Command	SpW	1553BC	RT, Receives Write Command
SpW Sends Read Command	SpW	1553BC	RT, Receives Read Command
SpW Sends Write Command	SpW	1394b	1394b Receives Write Quadlet Command
SpW Sends Write Command	SpW	1394b	1394b Receives Write Block Command
SpW Sends Read Command	SpW	1394b	1394b Receives Read Block Command
SpW Sends Read Response	SpW	1553RT	BC, Receives Read Response
SpW Sends Write Response	SpW	1553RT	BC, Receives Write Response
SpW Sends Read Response	SpW	1394b	1394b Receives Read Block Response
SpW Sends Write Response	SpW	1394b	1394b Receives Write Response
SpW Sends RMW Response	SpW	1394b	1394b Receives RMW Response
1394b Sends Write Quadlet Command	1394b	1553BC	RT, Receives Write Quadlet Command
1394b Sends Write Block Command	1394b	1553BC	RT, Receives Write Block Command
1394b Sends Read Block Command	1394b	1553BC	RT, Receives Read Block Command
1394b Sends Write Quadlet Command	1394b	SpW	SpW Receives Write Quadlet Command
1394b Sends Write Block Command	1394b	SpW	SpW Receives Write Block Command
1394b Sends Read Block Command	1394b	SpW	SpW Receives Read Block Command
1394b Sends Lock (RMW) Command	1394b	SpW	SpW Receives Lock (RMW) Command
1394b Sends Write Response	1394b	SpW	SpW Receives Write Response
1394b Sends Read Quadlet Response	1394b	SpW	SpW Receives Read Response

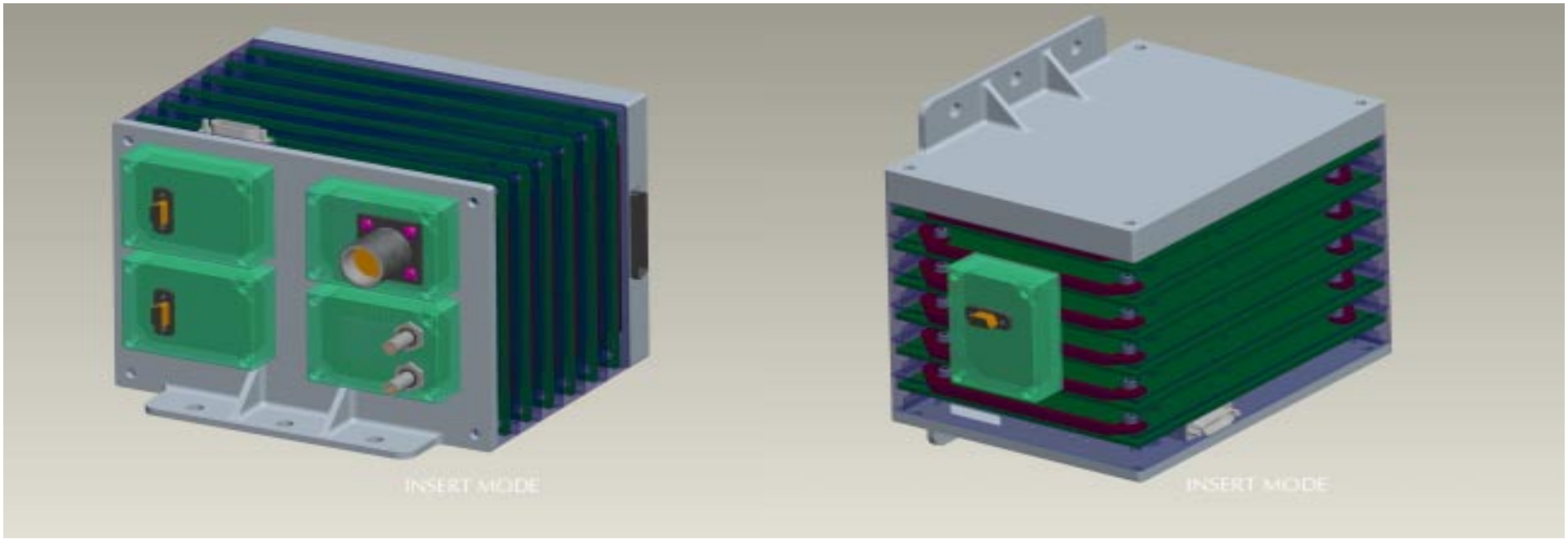
Future Work

- Implementation / testing of stream & burst modes
- Implementation of internal radiation mitigation techniques: i.e. Xilinx TMR, Actel scrubbing, and enhanced fault detection
- Full-up testing with real hardware (and not just testers)
  - Sequence of transactions at true speed
  - Illegal packet rejection
- Asynchronous stream packets – Better suited for Mil/Aerospace applications
- SAE AS5643 for “Mil-Std-FireWire” – Even better suited for Mil/Aerospace applications
- Auto-association of GUIDs with Node IDs
- Implement Time Triggered Ethernet (TTE)
  - Based on feasibility study performed under URTM NASA Langley contract
  - Increasingly popular in industry, mil & aerospace
  - Benefit to NASA Constellation I&T, missions



Future PIM Implementation

Repackage PIM design for use in spaceborne *reconfigurable* computer systems



Acknowledgements

This work was performed under contract to
the NASA Langley Research Center for the Radiation Hardened Electronics for Space Environments Reconfigurable Computing (RHESE RC) program.