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

Volume, Mass, Power

<table>
<thead>
<tr>
<th>Module</th>
<th>Mass (grams)</th>
<th>Mass (oz)</th>
<th>Power (Watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPTM</td>
<td>220</td>
<td>7.8</td>
<td>5.61</td>
</tr>
<tr>
<td>1553b PIM</td>
<td>75</td>
<td>2.6</td>
<td>0.132</td>
</tr>
<tr>
<td>SpaceWire PIM</td>
<td>65</td>
<td>2.4</td>
<td>0.337</td>
</tr>
<tr>
<td>1394 PIM</td>
<td>80</td>
<td>3.0</td>
<td>0.825</td>
</tr>
</tbody>
</table>

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
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

Master-Slave:
- **Multi-drop**
  - Bus Controller (Command)
  - 1553B
  - Remote Terminal (Response)
  - URTM

Peer-to-Peer:
- **Tree**
  - 1394b
  - URTM

Amorphous:
- **Point-to-Point Links**
  - SpaceWire
  - URTM

1394b/SpaceWire (remote terminals)
Protocol Overview

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

Data Link Conversion

- FireWire
- SpaceWire
- 1553B
- FireWire
Architecture Layers

Physical Layer | Data Link Layer | Abstraction Layer | Universal Bridge Logic and FSM

1553B PIM | 1553B BC Link Layer Controller | 1553B RT Link Layer Controller | Universal Message

SpaceWire PIM | SpaceWire Link Layer Controller | Message Type {Request, Response, Stream} | Message Subtype {Read, Write, Lock}

1394b PIM | 1394b Link Layer Controller | Destination {Address, Subaddress} | Payload {Data, Status}

Message Bus

Universal I/F

Address

Data Out

Data In

Configuration, Flow Control, and Status (CFCS)

Payload {Data, Status}

Universal Finite State Machine

Protocol-specific Input/Output Functions

Protocol Key Code

No Address Signals

Data Out (16)

Data In (16)

CFCS (8)

Address (16)

Data Out (16)

Data In (16)

CFCS (27)

Address (3)

Data Out (16)

Data In (16)

CFCS (40)

Address (22)

Data Out (16)

Data In (32)

CFCS (69)
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

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

### Actel FPGA

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Total:</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQUENTIAL</td>
<td>742</td>
<td>1080</td>
<td>68.70%</td>
</tr>
<tr>
<td>COMB</td>
<td>1432</td>
<td>1800</td>
<td>79.56%</td>
</tr>
<tr>
<td>LOGIC (seq+comb)</td>
<td>2174</td>
<td>2880</td>
<td>75.49%</td>
</tr>
<tr>
<td>IO w/ Clocks</td>
<td>52</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>CLOCK</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>HCLOCK</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

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

MAPLD 2009
Sigma Space Corporation
**Bilingual Dictionary Model**

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.

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

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

**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

1553B Command Word without sync or parity bits

1553 RT Link Layer Controller

Data Out (15:0)

Sync

Dat (15:0)

P

Address (15:0)

Sync

Address (15:0)

Command Word

Sync

RT Address

R

Subaddress

Data Word Count

P

Data  Words

again for each word

Number of bits

(3) (5) (1) (5) (5) (1)

Translates to

RMAP Write Request

Destination Logical Address

Protocol ID

Packet Type, Cmd, Srf Path Length

Destination Key

Source Logical Address

Transaction ID (MSB)

Transaction ID (LSB)

Extended Write Address

Data Length (MSB)

Data Length

Data Length (LSB)

Header CRC

Data

Data

Data

... transmitted last

Data CRC

EOP

Number of bits

(8) (8) (8) (8) (8)

Legend

Optional

Required

Variable Length Data

Automatic

Sigma Space Corporation

MAPLD 2009
Translation Mapping - Example

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

1553 RT Link Layer Controller

1553B Status Word (sync and parity automatically generated)
Status (9:0)

Source Logical Address
Source Path Address
Protocol ID
Packet Type, Cmd, Src Path Length
Status

Destination Logical Address
Transaction ID (MSB)
Transaction ID (LSB)
Reserved = 0

Data Length (MSB)
Data Length
Data Length (LSB)
Header CRC

Data
Data
Data
...

Data CRC
EOP

Legend:
Set Internally
Automatic
Optional
Required
Variable Length Data

Translates to
### URTM Testing

**Independent COTS Testers Validated Protocol Translation At Desired Rates**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Tester Description</th>
<th>Transmission Rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpaceWire</td>
<td>STAR-Dundee USB Brick &amp; Link Analyser</td>
<td>200 Mbits/sec Tx, 120 Mbits/sec Rx</td>
</tr>
<tr>
<td>1553B</td>
<td>Excalibur Systems PCMCIA Tester</td>
<td>1 Mbit / sec</td>
</tr>
<tr>
<td>1394b</td>
<td>DAP FireSpy</td>
<td>800 Mbits/sec</td>
</tr>
</tbody>
</table>
# Transactions Demonstrated

<table>
<thead>
<tr>
<th>Tester</th>
<th>Core</th>
<th>Core</th>
<th>Tester</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT, Sends Write Response</td>
<td>1553BC</td>
<td>SpW</td>
<td>Receives Write Response</td>
</tr>
<tr>
<td>RT, Sends Read Response</td>
<td>1553BC</td>
<td>SpW</td>
<td>Receives Read Response</td>
</tr>
<tr>
<td>RT, Sends Write Response</td>
<td>1553BC</td>
<td>1394b</td>
<td>Receives Write Response</td>
</tr>
<tr>
<td>RT, Sends Read Response</td>
<td>1553BC</td>
<td>1394b</td>
<td>Receives Read Response</td>
</tr>
<tr>
<td>BC, Sends Write Command</td>
<td>1553RT</td>
<td>SpW</td>
<td>Receives Write Command</td>
</tr>
<tr>
<td>BC, Sends Write Command</td>
<td>1553RT</td>
<td>1394b</td>
<td>Receives Write Command</td>
</tr>
<tr>
<td>SpW Sends Write Command</td>
<td>SpW</td>
<td>1553BC</td>
<td>RT, Receives Write Command</td>
</tr>
<tr>
<td>SpW Sends Read Command</td>
<td>SpW</td>
<td>1553BC</td>
<td>RT, Receives Read Command</td>
</tr>
<tr>
<td>SpW Sends Write Command</td>
<td>SpW</td>
<td>1394b</td>
<td>1394b Receives Write Quadlet Command</td>
</tr>
<tr>
<td>SpW Sends Read Command</td>
<td>SpW</td>
<td>1394b</td>
<td>1394b Receives Write Block Command</td>
</tr>
<tr>
<td>SpW Sends Write Command</td>
<td>SpW</td>
<td>1394b</td>
<td>1394b Receives Read Block Command</td>
</tr>
<tr>
<td>SpW Sends Read Response</td>
<td>SpW</td>
<td>1553RT</td>
<td>BC, Receives Read Response</td>
</tr>
<tr>
<td>SpW Sends Write Response</td>
<td>SpW</td>
<td>1553RT</td>
<td>BC, Receives Write Response</td>
</tr>
<tr>
<td>SpW Sends Read Response</td>
<td>SpW</td>
<td>1394b</td>
<td>1394b Receives Read Block Response</td>
</tr>
<tr>
<td>SpW Sends Write Response</td>
<td>SpW</td>
<td>1394b</td>
<td>1394b Receives Write Response</td>
</tr>
<tr>
<td>SpW Sends RMW Response</td>
<td>SpW</td>
<td>1394b</td>
<td>1394b Receives RMW Response</td>
</tr>
<tr>
<td>1394b Sends Write Quadlet Command</td>
<td>1394b</td>
<td>1553BC</td>
<td>RT, Receives Write Quadlet Command</td>
</tr>
<tr>
<td>1394b Sends Write Block Command</td>
<td>1394b</td>
<td>1553BC</td>
<td>RT, Receives Write Block Command</td>
</tr>
<tr>
<td>1394b Sends Read Block Command</td>
<td>1394b</td>
<td>1553BC</td>
<td>RT, Receives Read Block Command</td>
</tr>
<tr>
<td>1394b Sends Write Quadlet Command</td>
<td>1394b</td>
<td>SpW</td>
<td>SpW Receives Write Quadlet Command</td>
</tr>
<tr>
<td>1394b Sends Write Block Command</td>
<td>1394b</td>
<td>SpW</td>
<td>SpW Receives Write Block Command</td>
</tr>
<tr>
<td>1394b Sends Read Block Command</td>
<td>1394b</td>
<td>SpW</td>
<td>SpW Receives Read Block Command</td>
</tr>
<tr>
<td>1394b Sends Lock (RMW) Command</td>
<td>1394b</td>
<td>SpW</td>
<td>SpW Receives Lock (RMW) Command</td>
</tr>
<tr>
<td>1394b Sends Write Response</td>
<td>1394b</td>
<td>SpW</td>
<td>SpW Receives Write Response</td>
</tr>
<tr>
<td>1394b Sends Read Quadlet Response</td>
<td>1394b</td>
<td>SpW</td>
<td>SpW Receives Read Response</td>
</tr>
</tbody>
</table>
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.