The basic motivation is to provide an open, generic and fully integrated protocol validation system (PVS) for satellite on-board communications supporting multiple physical interfaces (SpW, MIL-STD-1553) and functionalities (emulation, validation, interworking testing, monitoring).

- Current evolution of satellite on-board communications, require the development & experimentation with new dedicated communication protocols and services (SpW, SOIS, etc.)
- New generation of validation tools is required to support advanced protocol development, test, integration & validation
- A protocol validation tool with more than 20 years of experience in the telecommunication sector. & with hundreds of installations worldwide
- Has been widely used for testing various telecommunication networks (ISDN, V5, SS7, IN, GSM, UMTS, VoIP, custom)
PVS at a glance

- Multiple Networks support (SpW,1553, CAN)
- Rapid Prototyping
- SAFIRE TOOLCHAIN
- CONFORMANCE TESTING
- Interworking & Interoperability Testing
- EMULATION
- Protocol & Device Emulation
- TRAFFIC GENERATION
- Fault Injection
- NETWORK MONITORING
- Stress Testing
- Functional Testing

Esa - EADS Astrium - Teletel
PVS foreseen features

- **DEVICE EMULATION**: economic & portable replacement of a network element in the testbed

- **PROTOCOL EMULATION**: experimentation with various protocol features (parameterization of protocol variables, exclusion/inclusion of protocol optional functions, combination of multiple protocols)

- **CONFORMANCE TESTING**: execution of tests to ensure that a device (System Under Test) is operating in compliance with the applicable ECSS and CCSDS standards.

- **FAULT-INJECTION**: injection of errors at various protocol layers to validate the response of the devices/networks in erroneous conditions

- **TRAFFIC GENERATION**: generation of traffic for validation of higher layer protocols or bulk traffic injection at lower layers for performance evaluation and network dimensioning

- **NETWORK MONITORING**: network monitoring, through direct physical traffic acquisition (network statistics, error detection, troubleshooting)
Current contract technical objectives

- **PVS Phase 1 (Feb 2009 – Jan 2010):**
  - Requirements capturing & analysis, based on requirements by ESA and EADS Astrium, and top level partitioning
  - Technology review on related technologies, tools and protocols
  - Identification of SpW-T features to validate
  - Realisation of a PVS proof-of-concept prototype for SpW networks
  - Evaluation and demonstration of the PVS with SpW-T and GAMMA protocols
  - Development plan definition for the full PVS
Results: Hardware platform

- 4 SpW ports
- FPGA protection
- Fine (KHz) Tx clock granularity
- trigger I/F
- > 300 Mbps SpW Line Rate
Results: Integration with SAFIRE graphical tool chain
Results: Validation of SpW-T and GAMMA protocols
PVS Phase 1 System Architecture

Host Platform
- Protocol Libraries
- Higher Layer SW (PVS Control)
- PLDA Driver Adaptations & Extensions (SpWSpW-T Driver)
- PLDA PCI Driver

High performance acceleration board (COTS)
- SpWSpW-T FPGA
  - 2 x SpW-T v3.1 Blocks
  - 2 x SpaceWire (ECSS-E-ST-60-12C)

FPGA Logic
- SPI Controller
- Control/Status registers
- PLDA PCIe Gen 2 Core
- PLDA EzDMA Core
- Rx Arbiter
- Tx Arbiter

- Embedded Processor
- High Gate Count FPGA
- On board memory (612 MBytes)
- LVDS Xeivers (SpW signal Level)

SpW AoB
- LVDS Xeivers
- Programmable Oscillator

esa
EADS Astrium
teletel
✓ Segmentation
✓ End to End flow Control
  X SBFCT support
  X BFCT Timeout/Retransmission
✓ Acknowledgement
✓ Address Translation
  X Path addressing
✓ PDU Encapsulation
✓ Resource Reservation
✓ Error Detection
  ✓ Header/data CRC
  ✓ Sequence Number
  ✓ Missing ACK
<table>
<thead>
<tr>
<th>Function</th>
<th>Registers</th>
<th>LUTs</th>
<th>Slices</th>
<th>BRAMs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segmentation</td>
<td>106</td>
<td>197</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>Tx Encapsulation</td>
<td>671</td>
<td>1182</td>
<td>552</td>
<td>8</td>
</tr>
<tr>
<td>Tx Acknowledgement</td>
<td>619</td>
<td>837</td>
<td>35</td>
<td>1</td>
</tr>
<tr>
<td>Tx Flow Control</td>
<td>50</td>
<td>124</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>Resource Reservation</td>
<td>354</td>
<td>325</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Rx Encapsulation</td>
<td>400</td>
<td>341</td>
<td>295</td>
<td>1</td>
</tr>
<tr>
<td>Rx Acknowledgement</td>
<td>70</td>
<td>141</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td>Rx Flow Control</td>
<td>64</td>
<td>123</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>Reassembly (Logic)</td>
<td>170</td>
<td>164</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>Reassembly (Buffers)</td>
<td>588</td>
<td>412</td>
<td>464</td>
<td>4</td>
</tr>
<tr>
<td>Tx Statistics</td>
<td>18</td>
<td>86</td>
<td>35</td>
<td>1</td>
</tr>
<tr>
<td>Rx Statistics</td>
<td>22</td>
<td>259</td>
<td>116</td>
<td>3</td>
</tr>
<tr>
<td><strong>SpW-T Block</strong></td>
<td>3313</td>
<td>4639</td>
<td>2405</td>
<td>18</td>
</tr>
<tr>
<td><strong>SpW-b Core</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpW-b Core</td>
<td>520</td>
<td>528</td>
<td>383</td>
<td>2</td>
</tr>
<tr>
<td><strong>Tx DMA Arbiter</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tx DMA Arbiter</td>
<td>621</td>
<td>677</td>
<td>337</td>
<td></td>
</tr>
<tr>
<td>Rx DMA Arbiter</td>
<td>540</td>
<td>964</td>
<td>363</td>
<td></td>
</tr>
<tr>
<td>Pointers Bank</td>
<td>204</td>
<td>147</td>
<td>110</td>
<td>2</td>
</tr>
<tr>
<td><strong>PVS with 2 x SpW/SpW-T</strong></td>
<td>20786</td>
<td>26942</td>
<td>13360</td>
<td>64</td>
</tr>
</tbody>
</table>
- PVS SpW-T tested against SpW-T SW implementation on Linux 2.6 using 4-Links DSI
- 4Links FSR router
- Monitoring through Star-Dundee IP Tunnel
- Remote integration tests through internet
- Same tests executed in remote & local configurations
- Endurance testing executed on Scheduled mode with transfers on more than 12 hours (65 GB logfile)
## SpW-T Test Results

<table>
<thead>
<tr>
<th>Description</th>
<th>Error injected</th>
<th>Error detected</th>
<th>Verdict</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal asynchronous/Scheduled communication</td>
<td>-</td>
<td>-</td>
<td>PASS</td>
</tr>
<tr>
<td>Asynchronous/Scheduled communication with error</td>
<td>SQ</td>
<td>YES</td>
<td>PASS</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>YES</td>
<td>PASS</td>
</tr>
<tr>
<td></td>
<td>HDR CRC</td>
<td>YES</td>
<td>PASS</td>
</tr>
<tr>
<td></td>
<td>Data CRC</td>
<td>YES</td>
<td>PASS</td>
</tr>
<tr>
<td>Asynchronous communication with missing ACK</td>
<td>ACK inhibit</td>
<td>YES</td>
<td>PASS</td>
</tr>
<tr>
<td>Asynchronous communication with invalid ACK</td>
<td>CH</td>
<td>YES</td>
<td>PASS</td>
</tr>
<tr>
<td></td>
<td>SQ</td>
<td>YES</td>
<td>PASS</td>
</tr>
<tr>
<td></td>
<td>CRC</td>
<td>YES</td>
<td>PASS</td>
</tr>
<tr>
<td>Asynchronous communication without congestion</td>
<td>-</td>
<td>-</td>
<td>PASS</td>
</tr>
<tr>
<td>Asynchronous communication with congestion</td>
<td>-</td>
<td>-</td>
<td>PASS</td>
</tr>
<tr>
<td>Scheduled communication without congestion</td>
<td>-</td>
<td>-</td>
<td>PASS</td>
</tr>
<tr>
<td>Scheduled communication with congestion</td>
<td>-</td>
<td>-</td>
<td>PASS</td>
</tr>
</tbody>
</table>
SpW-T V3.1 specification/implementation issues

- Error handling is restricted to data errors
- Timing errors are not addressed (e.g. Time Code loss)
- Action to perform in case of error at the level of application is not defined
- Problem with BFCTs during initialization. If destination sends BFCTs while source is not ready the BFCT is lost. The destination shall retry until BACK is received. How many times?
- The SBFCT time constraint (3 us in the example of the V3.1 spec.) not realistic for SW implementation & requires high speed HW operation (> 100MHz)
- Need to access the Token buffer through the application (e.g. in case of PDU loss the BFCT is consumed and never received from the remote side)
- SQ storage at various functions requires many memory resources
- The need for separate UDS buffers at the receiver increases memory needs even more
- Other minor issues (e.g. values not specified for DP, ACK/SACK, BFCT/SBFCT, BACK)
Issues to be considered on the next spec. revision - Acknowledgement (1/2)

For the acknowledgement function the handling of (S)ACKs when multiple SpW-T channels are used, forced us to implement a common Timeout FIFO, which significantly complicates the design of the “timer invalidation” block in order to:

- Handle out-of-order ACKs
- Compensate for ACK losses
Issues to be considered on the next spec. revision - Acknowledgement (2/2)

Timer Invalidation logic cannot delete the R, SQR entry. It shall:
1) buffer the R, SQR event,
2) Wait for N, SQN timeout
3) Wait for P, SQP (S)ACK or timeout
4) Invalidate the R, SQ entry
5) Clear the R, SQR event
**Problem**: The combination of {Destination Address, Channel ID, Source Address} do not form a contiguous address space

- Look up cannot be performed to associate this combination with a certain flow

- The search shall be performed by several SpW-T functions (Reassembly, Acknowledgement, Flow Control)

**Alternatives**: A Classifier block, replacing the {Destination Address, Channel ID, Source Address} combination with a FlowID was developed

- CAM based: Expensive, slows down overall performance

- Linear search: Slow, not scalable

- Binary search: Scalable but more complex

Issues to be considered on the next spec. revision – Channels handling
Conclusions

- PVS/DSI is among the first validated SpW-T implementations

- Current specification (v3.1) has several open issues

- Next specification revision shall heavily consider:
  - Implementation issues (!)
  - Error handling at application level
Antonis Tavoularis, Vangelis Kollias
TELETEL SA
A.Tavoularis@TELETEL.eu
V.Kollias@TELETEL.eu
www.teletel.eu

Christophe Honvault
EADS Astrium SAS Satellites
christophe.honvault@astrium.eads.net
www.astrium.eads.net