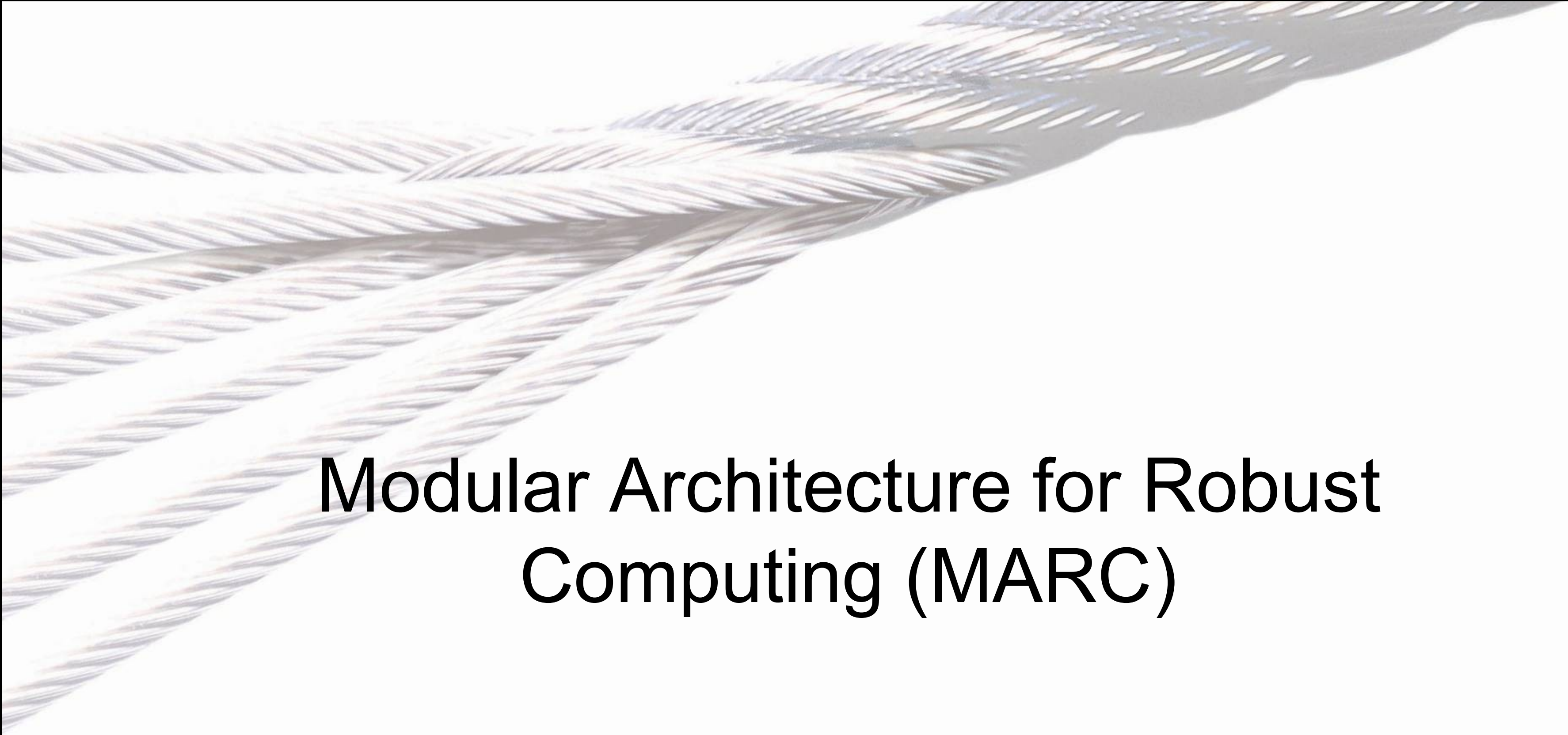


Modular Architecture for Robust Computing (MARC)

Presented by Alan Senior

17th September 2007

at the

International SpaceWire Conference 2007

S·E·A

Project team

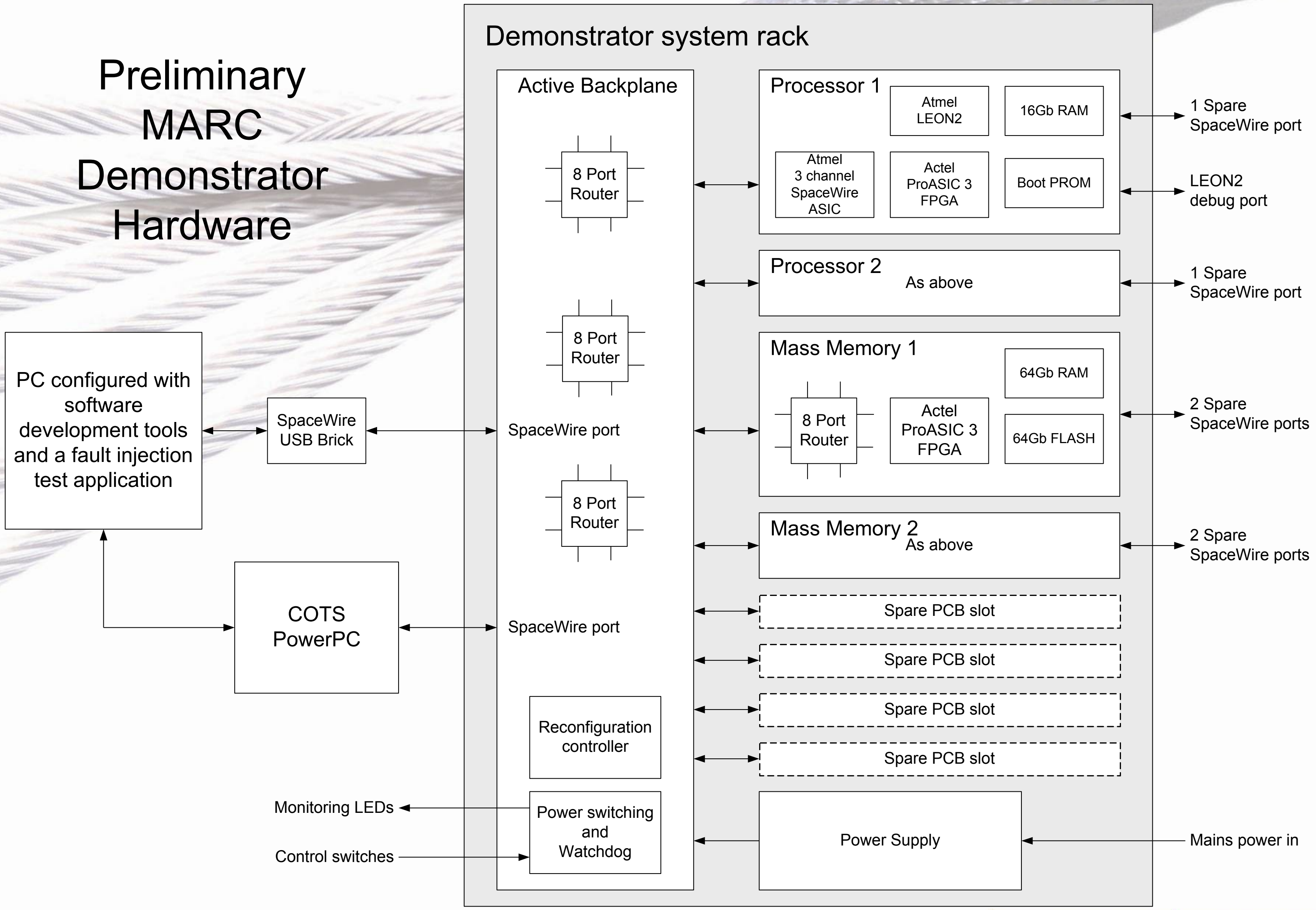
- Systems Engineering & Assessment Ltd (SEA)
  - Definition of the system hardware architecture
  - Selection of component technologies
  - Design, manufacture and test of a demonstrator system
  - Implement hardware FDIR functions
- SciSys UK Ltd
  - Develop the Generic Fault-tolerant Architecture using SOIS (GenFAS) software framework
  - Implement software FDIR functions
- EADS Astrium Ltd
  - Definition of FDIR requirements, partitioning and algorithms
  - System analysis and verification of demonstrator performance
  - Activity coordination

MARC Aims

- Define a modular architecture based on a SpaceWire network that is scalable to meet future mission needs
- Design, manufacture and test a representative demonstration system comprising:
  - Processing modules, Mass Memory (RMAP interface) and active SpW backplane
  - New flight capable hardware technologies (eg LEON2, SpW router)
  - COTS Power PC to demonstrate network load handling
  - SOIS based software and related services to ECSS-E-40
- Demonstrate the essential features of a heterogeneous, fault tolerant, high availability distributed avionics system

S·E·A

Preliminary MARC Demonstrator Hardware



Demonstrator system rack

Active Backplane

8 Port Router

8 Port Router

SpaceWire port

8 Port Router

SpaceWire port

Reconfiguration controller

Power switching and Watchdog

Processor 1

Atmel LEON2

16Gb RAM

Atmel 3 channel SpaceWire ASIC

Actel ProASIC 3 FPGA

Boot PROM

1 Spare SpaceWire port

LEON2 debug port

Processor 2

As above

1 Spare SpaceWire port

Mass Memory 1

64Gb RAM

8 Port Router

Actel ProASIC 3 FPGA

64Gb FLASH

2 Spare SpaceWire ports

Mass Memory 2

As above

2 Spare SpaceWire ports

Spare PCB slot

Spare PCB slot

Spare PCB slot

Spare PCB slot

Power Supply

Mains power in

PC configured with software development tools and a fault injection test application

SpaceWire USB Brick

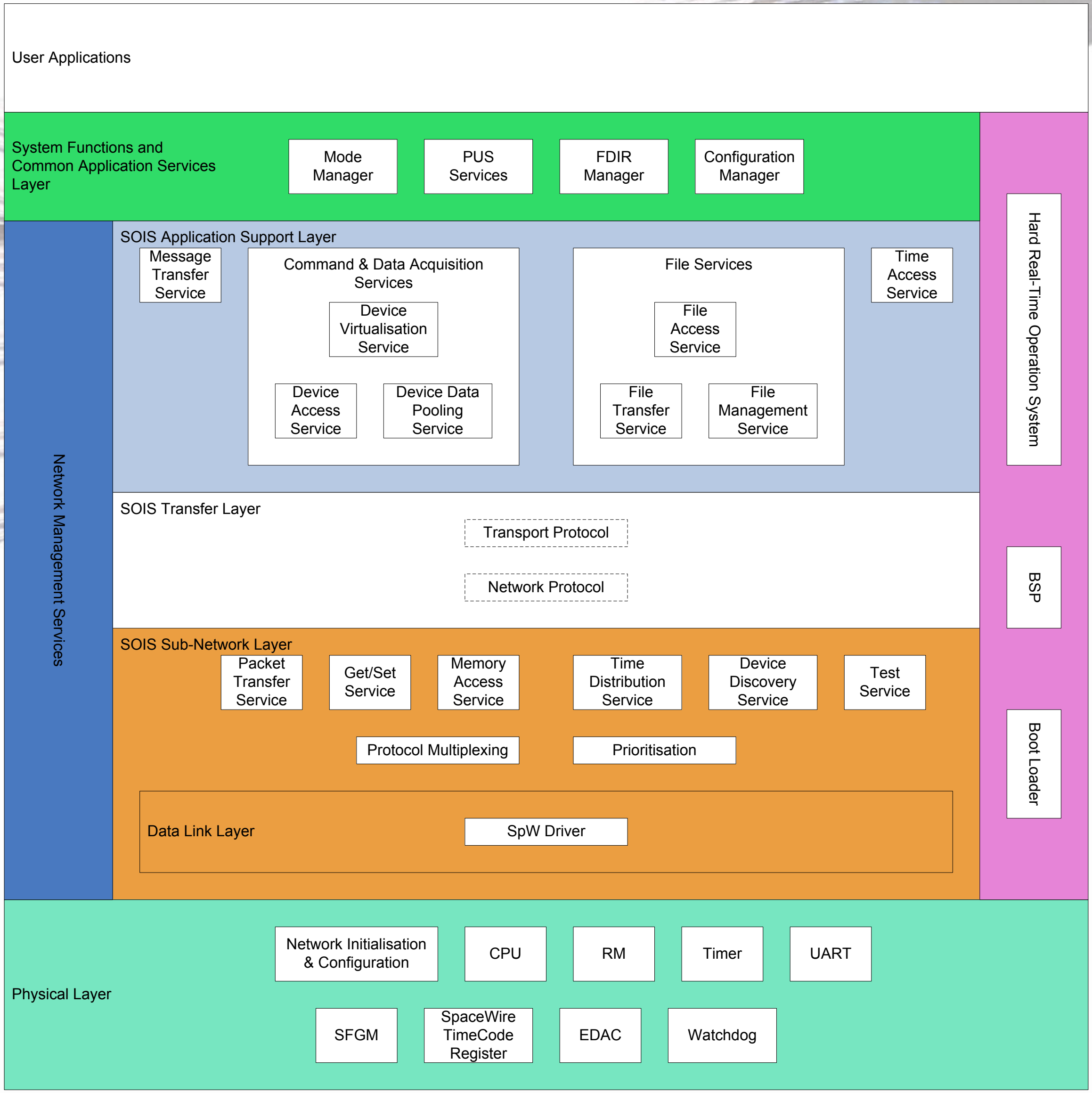
COTS PowerPC

Monitoring LEDs

Control switches

GenFAS features

- Decentralised, distributed Onboard software
  - May be located on any processor
  - Telecommands routed based on APIDs
- Decentralised, distributed access to all instruments and transducers via a SpaceWire network
- Advanced Mass Memory architecture
  - Supporting a file system, packet store, FIFO buffer etc.
- Fault tolerance based on “spare” capability, rather than full redundancy
  - Applications are linked together into a validated software build
  - Build information stored in a safeguarded context area
  - Software builds are allocated by a configuration manager to processors
  - FDIR manager detects failures and initiates re-deployment of the software build to a spare processor



User Applications

System Functions and Common Application Services Layer

Mode Manager

PUS Services

FDIR Manager

Configuration Manager

Hard Real-Time Operation System

BSP

Boot Loader

Network Management Services

SOIS Application Support Layer

Message Transfer Service

Command & Data Acquisition Services

Device Virtualisation Service

Device Access Service

Device Data Pooling Service

File Services

File Access Service

File Transfer Service

File Management Service

Time Access Service

SOIS Transfer Layer

Transport Protocol

Network Protocol

SOIS Sub-Network Layer

Packet Transfer Service

Get/Set Service

Memory Access Service

Time Distribution Service

Device Discovery Service

Test Service

Protocol Multiplexing

Prioritisation

Data Link Layer

SpW Driver

Physical Layer

Network Initialisation & Configuration

CPU

RM

Timer

UART

SFGM

SpaceWire TimeCode Register

EDAC

Watchdog

Software architecture

FDIR analysis and performance

- Define failure scenarios, detection strategies and levels of autonomy
- Partition FDIR functions into onboard hardware and software or TM/TC actions
- Define FDIR architecture as centralised (single node), distributed FDIR (multiple nodes with majority voting) or a combination of both
- Create an FDIR analysis tool in UML to derive the FDIR actions
- Create an FDIR model of the system incorporating the FDIR algorithm using UML
- Evaluate and optimise the system architecture performance when exposed to different failure scenarios
- Demonstrate the FDIR algorithm running on the MARC demonstrator, handling failures
- Assess the performance and reliability of the SpaceWire network to handle critical commands/telemetry and hence act as an alternative to Mil-Std-1553B