OpenComRTOS:

Formally developed RTOS for Heterogeneous Systems

Bernhard H.C. Sputh, Eric Verhulst, and Vitaliy Mezhuyev
Email: {bernhard.sputh, eric.verhulst, vitaliy.mezhuyev}@altreonic.com

http://www.altreonic.com

From Deep Space to Deep Sea

Push Button High Reliability

Outline

- History of Altreonic
- OpenComRTOS Information
- Performance Figures
- Conclusions

History of Altreonic

- Eonic (Eric Verhulst): 1989 – 2001
  - Developed Virtuoso a Parallel RTOS (sold to Wind River Systems);
  - Communicating Sequential Processes as foundation of the “pragmatic superset of CSP”;
- Open License Society: 2004 – now
  - R&D on Systems and Software Engineering;
  - Developed OpenComRTOS using Formal Methods
- Altreonic: 2008 – now
  - Commercialises OpenComRTOS;
  - Based in Linden (near Leuven) Belgium;

OpenComRTOS Factsheet 1

- CSP inspired Real-time Operating System;
- Formally designed and developed;
- Small code size (typically 5 – 10 KiB);
- Network centric;
- Support for Heterogeneous Systems;
- Currently available Ports:
  - Posix32 and Win32
  - ARM Cortex M3
  - Leon3, Xilinx Microblaze, MLX-16
  - XMOS (experimental, under development)

OpenComRTOS Factsheet 2

- OpenComRTOS has two principal components:
  - Tasks:
    - Prioritised (256 priorities are available);
    - Tasks do not share memory;
    - Tasks communicate with Hubs using Packets.
    - Also the Kernel is a task
  - Hubs:
    - Generic synchronisation primitive in OpenComRTOS
    - Hubs operate system-wide, but transparently → Virtual Single Processor (VSP) programming model.

Services offered by Hubs

- Event – Synchronisation on a Boolean value;
- Semaphore – Synchronisation with a counter;
- Port – Synchronisation and exchanging a packet, i.e. data transport (CSP Channel like);
- Resource – Locking mechanism, with ownership;
- FIFO – Buffered packet communication;
- Memory Pool – For the allocation of memory;

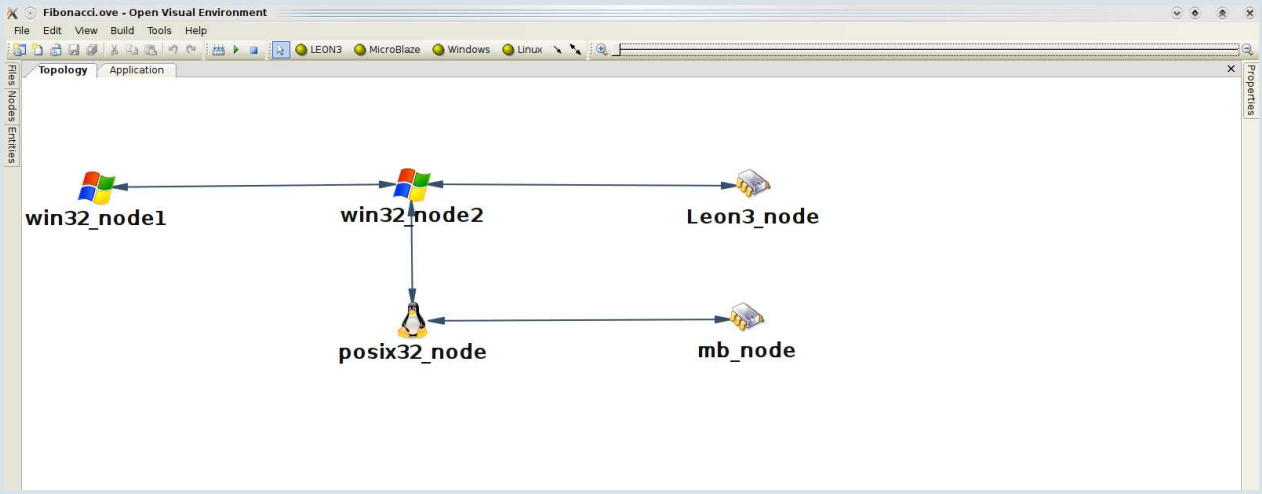
Types of Hub-Interactions

- Interactions can be of four different types in OpenComRTOS:
  - _NW – Non waiting
  - _W – Waiting
  - _WT – Waiting with timeout
  - _A – Asynchronous

Virtual Single Processor programming Model

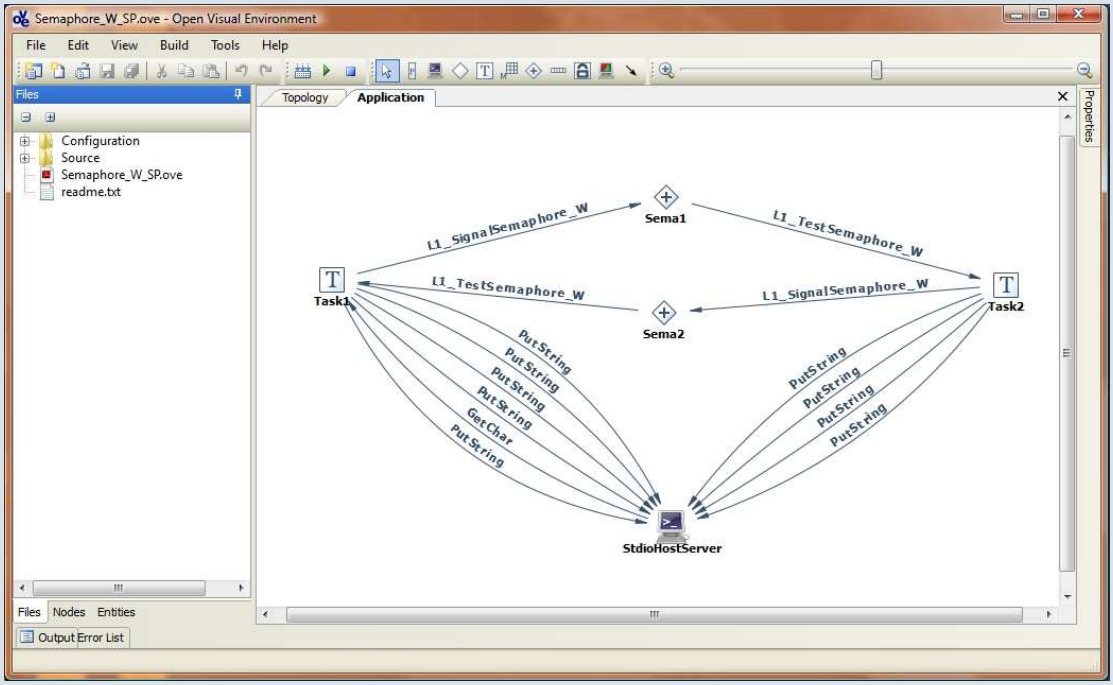
- This means: “Transparent access to any entity in the system.”
- This is possible due to the separation of Topology and Application.
  - Topology: Defines the Processing Entities (Nodes) and the communication Links between them.
  - Application: Tasks, Hubs and their interactions.

VSP Workflow 1



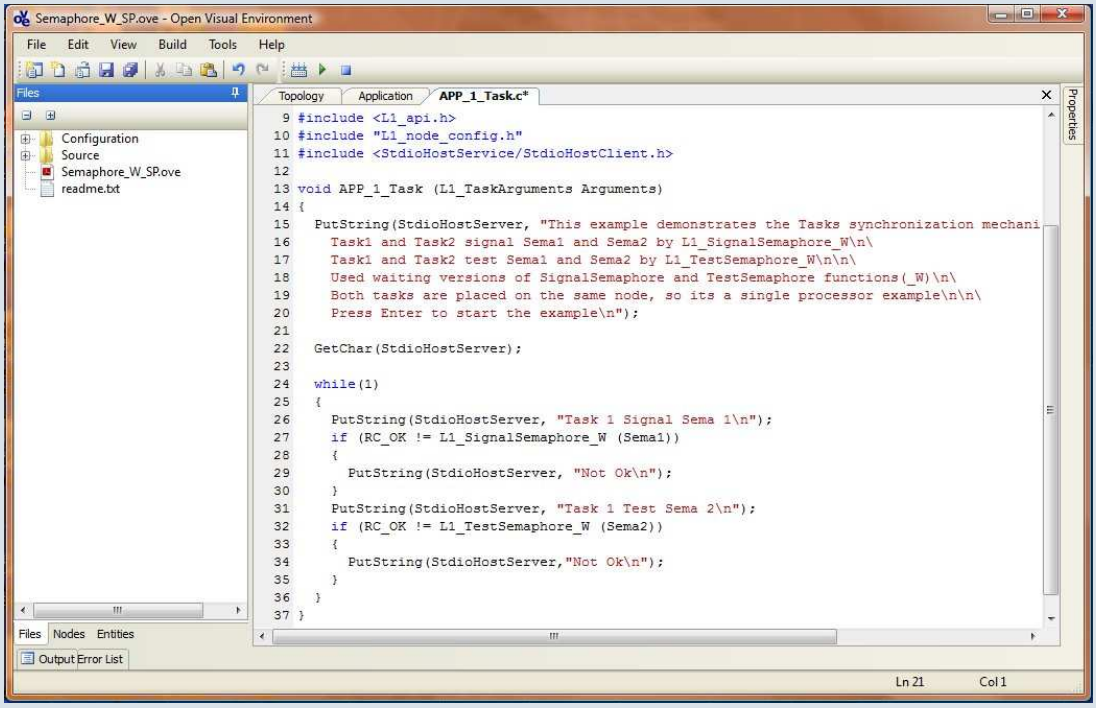
Define the Topology using OpenVE

VSP Workflow 2



Define the Application using Tasks, Hubs, and Interactions

VSP Workflow 3



Define the behaviour of the tasks using C.

VSP Workflow 4

- Build the project, this will:
  - Generate the necessary routing tables and allocate all necessary variables. Thus no runtime allocation necessary → One major source of errors resolved.
  - Generate the Build system configuration for each Node in the system. Independent of its type
  - Compile and link each Node and install the resulting binaries in a central location.
  - The resulting binaries can now be downloaded to the nodes and exectued.

Performance of OpenComRTOS

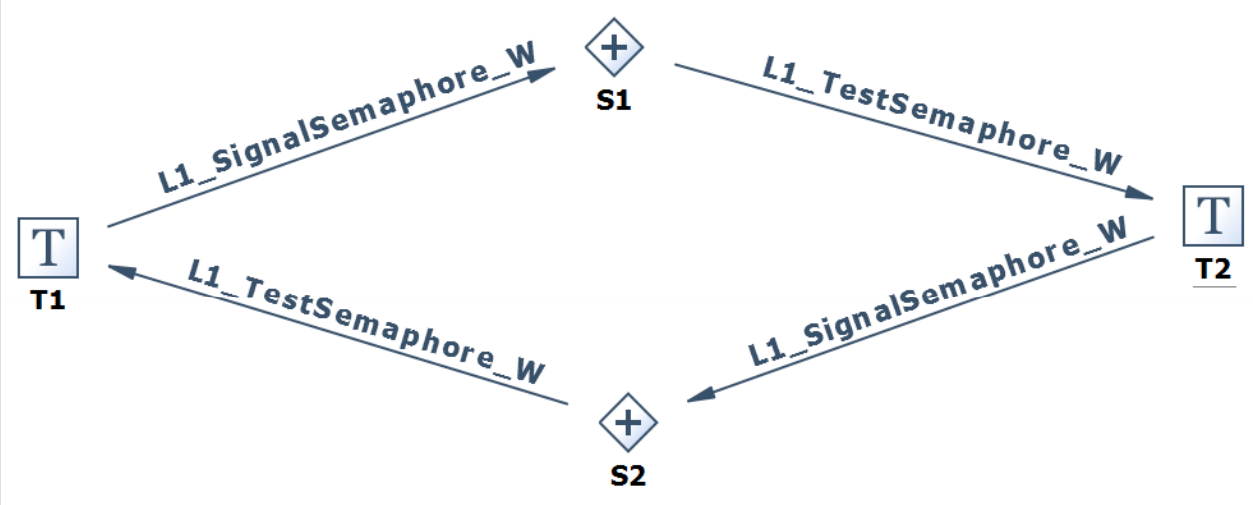
- Code size figures for different targets
- Context Switch Measurements
- Interrupt Latency on an ARM Cortex M3

Codesize Figures

Service	MLX-16	MicroBlaze	Leon3	ARM	XMOS
L1 Hub shared	400	4756	4904	2192	4854
L1 Port	4	8	8	4	4
L1 Event	70	88	72	36	54
L1 Semaphore	54	92	96	40	64
L1 Resource	104	96	76	40	50
L1 FIFO	232	356	332	140	222
L1 PacketPool	NA	296	268	120	166
Total L1 Services	1048	5692	5756	2572	5414

Code size figures (in Bytes) obtained for our different ports, compiled with Optimisation Os

Semaphore Loop Performance



To calculate the loop time the time for 1000 iterations gets taken, using the highest precision timer available in the system

Measured Loop Times

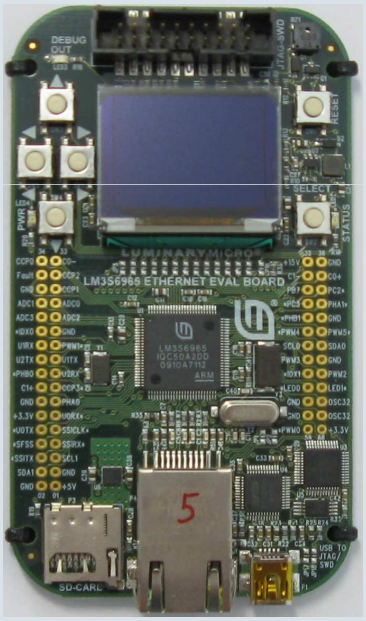
	Clock speed	Context size	Memory location	Loop time
MLX-16	6MHz	4 x16bit	internal	100.8µs
Xilinx Microblaze	100MHz	32 x 32bit	internal	33.6µs
Leon3	40MHz	32 x 32bit	external	136.1µs
ARM	50MHz	16 x 32bit	internal	52.7µs
XMOS	100MHz	14 x 32bit	internal	26.8µs

Interrupt Latency Measurement

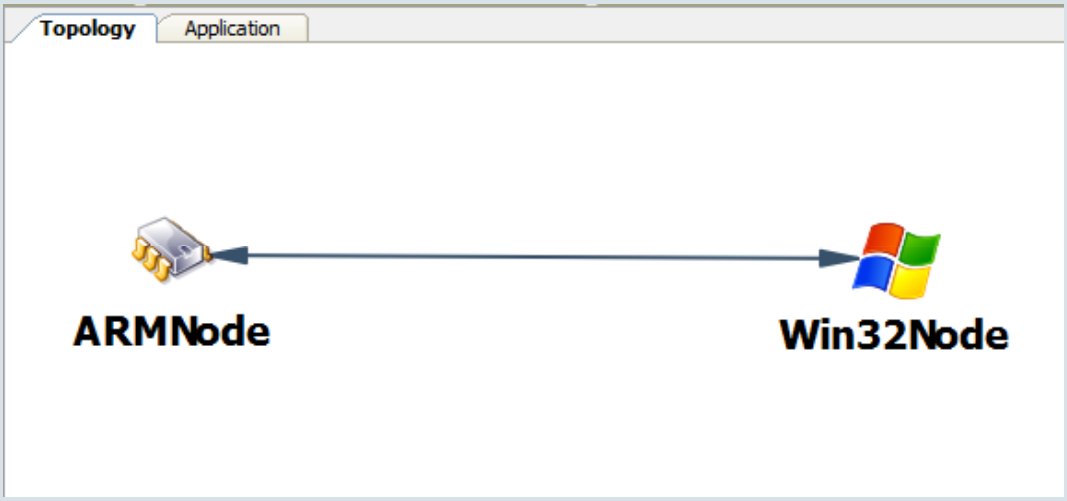
- Interrupt Latency is the time after an Interrupt Request (IRQ) occurred, until this message gets passed to a certain instance in the System.
- In OpenComRTOS we differentiate between two different Interrupt Latencies:
  - IRQ to ISR Latency
  - IRQ to Task Latency

Hardware Setup

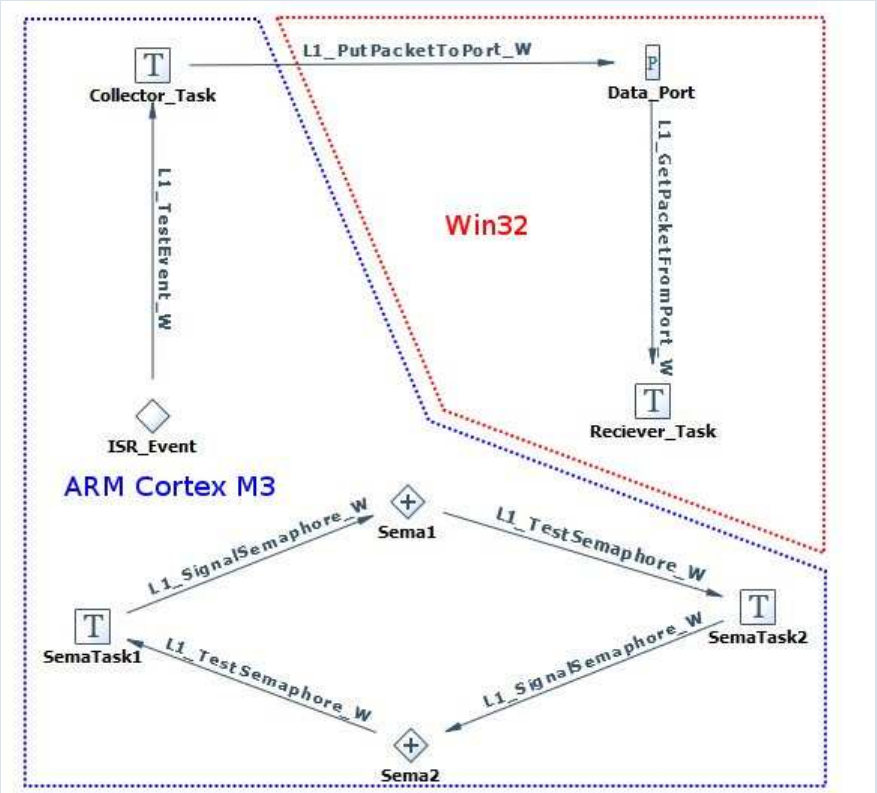
- Device under Test:
  - CPU: 50MHz ARM Cortex-M3 (LM3s6965)
  - 64kB RAM
  - 256kB Flash
  - 50MHz timer / counter



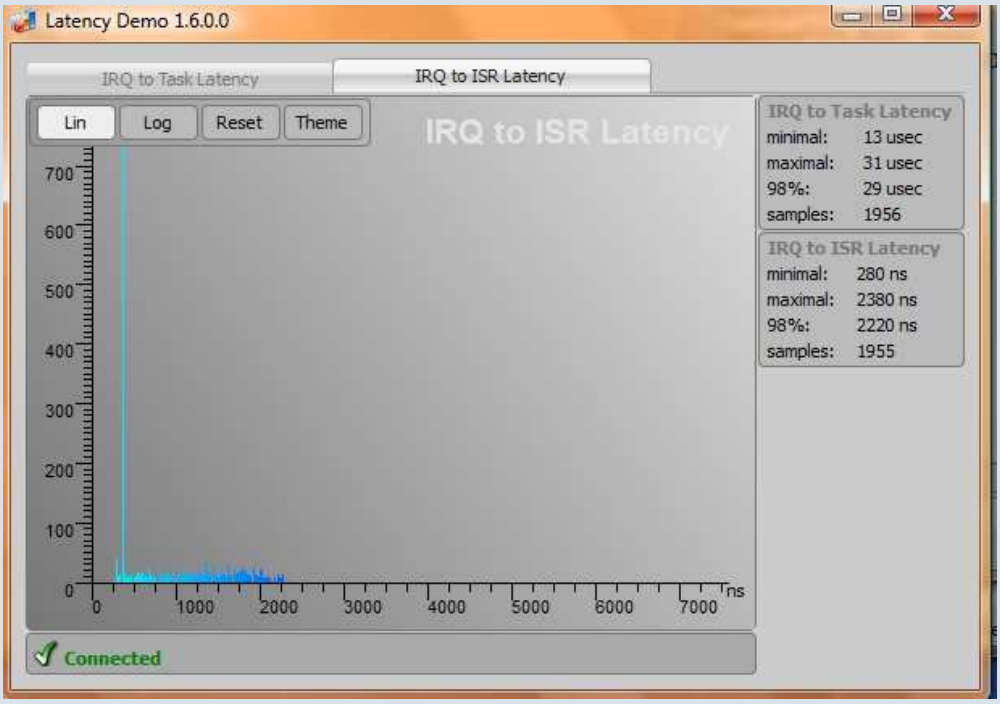
System Topology



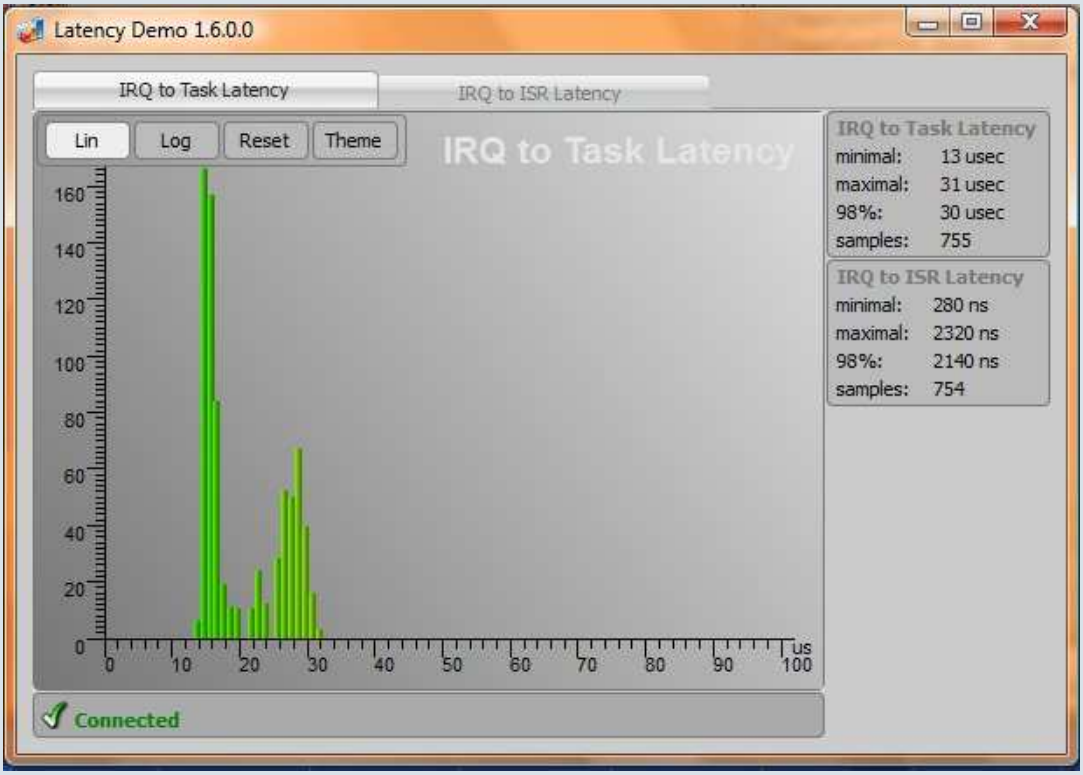
System Application Diagram



Measured IRQ to ISR Latency



Measured IRQ to Task Latency



Conclusions

- Advantages of the formal development:
  - Small code size (~10x smaller than Virtuoso from Eonic);
  - Higher Performance, due to less code to be executed;
  - Highly portable;
- Separation of topology and application resulted in a Virtual Single Processor (VSP) programming model.
- No failing malloc-operations, due to static allocation of memory during the build process.
- Support for systems consisting of different processor architectures.
- Supports for systems that use different communication technologies to represent Links.

Questions?

Thank You for your attention

Visit us in Hall 11.0 Booth 102

Altreonic

“If it doesn't work, it must be art.
If it does, it was real engineering”