

Transparent Programming of Many/Multi Cores with OpenComRTOS

Comparing Intel 48-core SCC and TI 8-core TMS320C6678

Bernhard H.C. Spath, Andrew Lukin, and Eric Verhulst
Altreonic NV

bernhard.spath, eric.verhulst@altreonic.com

Trustworthy Forever

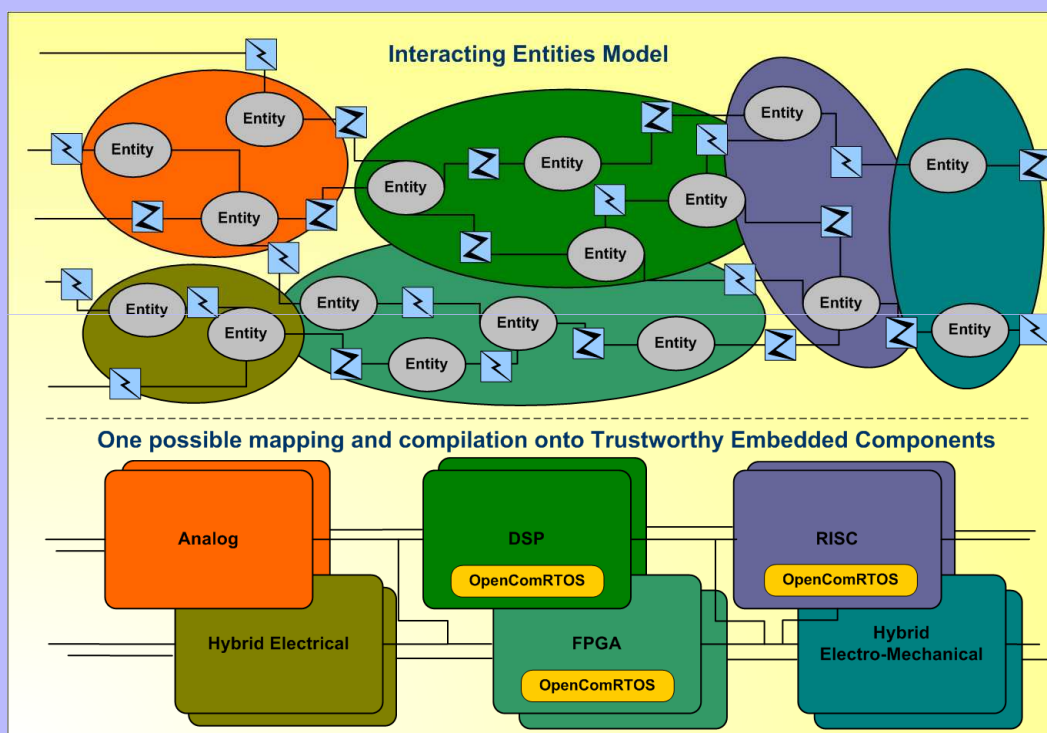
From Deep Space To Deep Sea

Company profile

- History goes back to 1989 (**Eonic Systems**)
 - **VIRTUOSO** parallel RTOS (T800, C40, C6x, 2016x, TS102, G4, ...) – **CSP based**
 - Used from 1 CPU to 1600 DSPs (sonar, radar) to 12000 nodes (heterogeneous)
 - Acquired by Wind River Systems in 2001
- **Altreonic**: created as spin-off in 2008 following R&D
 - Unified systems engineering (**GoedelWorks**)
 - Formalised when possible
 - Network-centric **OpenComRTOS**:
 - Used as test case for use of formal techniques
 - Binary/source and Open Technology License model

- Computation to communication ratio:
 - Depends on application
 - Input rate = output rate = $\frac{1}{2}$ computation rate ideal
 - Ideal ratio = 1, typically minimum 5 to 10
- Data communication is bottleneck:
 - Set-up latency, no polling
 - Task to task/memory to memory bandwidth is real target
 - Concurrency to mask communication latencies
 - DMA => requires additional busses
- Data protection:
 - Pointers are fast but very dangerous
 - memcpy has no distributed semantics! (should be _W)
- Heterogeneous many-cores: data-types

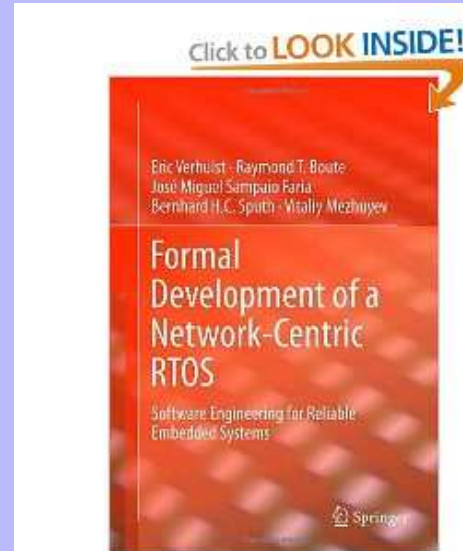
The OpenComRTOS Goal: program once, run anywhere



The OpenComRTOS Project



- Novel programming model, but long formal history (Hoare's CSP, 1975)
- Use of TLA+/TLC for design and verification
- Unexpected result:
 - 10X smaller code size;
 - Easy to Port to new Platforms;
 - Low amount of assembly;



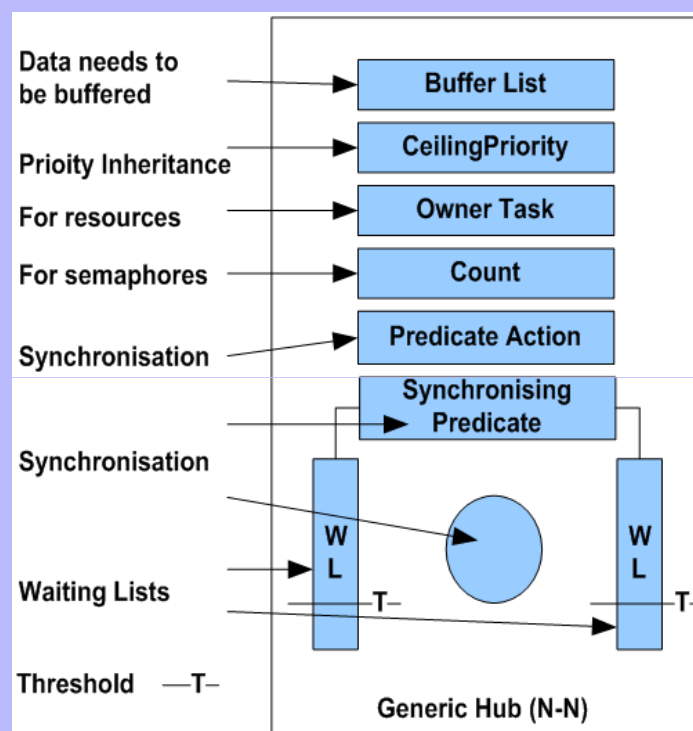
OpenComRTOS Properties



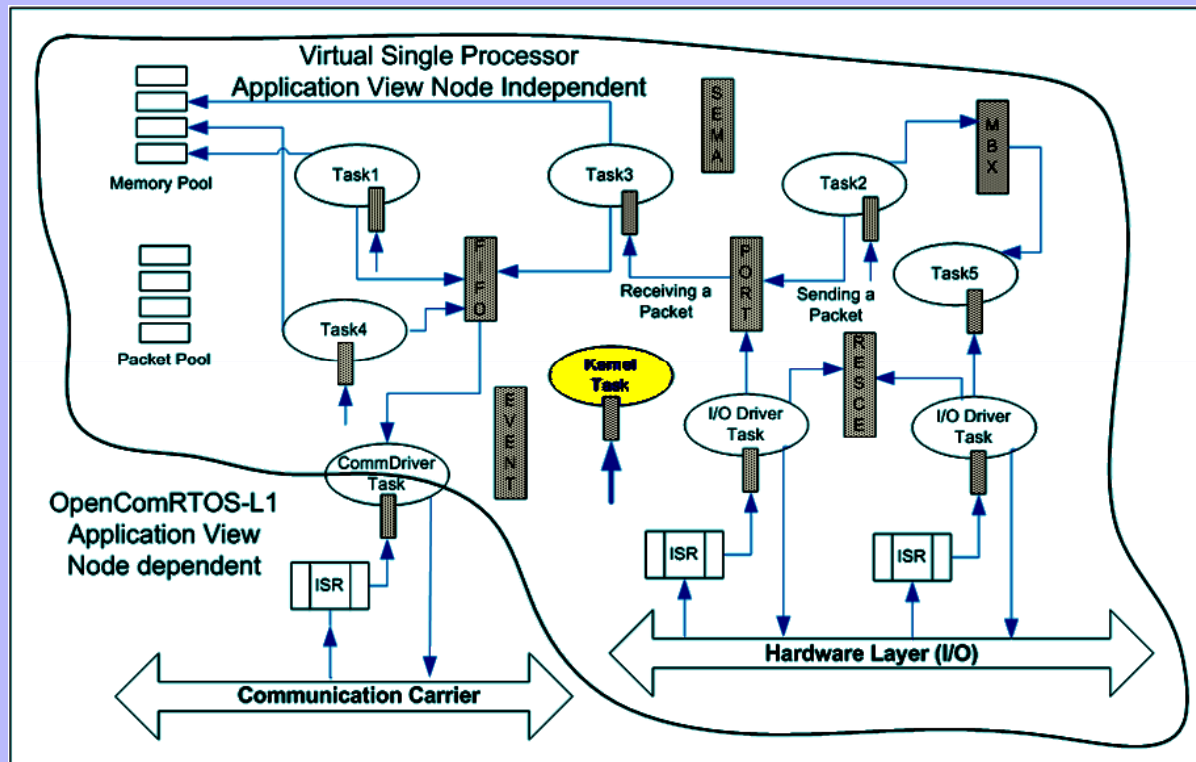
- Network-centric (RT)OS, MP by default:
 - Concurrency at the core (“Interacting Entities”);
 - Pragmatic superset of CSP (Hoare);
 - Scalable yet very small: typically 2 to 5 kiB/node;
 - Real-time communication as system level service;
 - Unique support for distributed priority inheritance;
 - Heterogeneous target /communication support;
 - Integrate seamlessly “legacy OS” nodes;
 - Virtual Single Processor model;
 - Visual modeling/ programming with code generators;
 - Capable of fault-tolerance and resource management.

- Basis:
 - RMA => priority based, preemptive
 - Additional: timer based
 - Everything is priority ordered
 - e.g. waiting lists
 - Packet based communication
- Priority Inheritance support with ceiling level
 - Single processor
 - Unique distributed implementation

The Generic Hub as metamodel



Similar to a Guarded Atomic Action, or a pragmatic superset of CSP



Interaction semantics

Hub Entity	Semantics
Event	Synchronisation on boolean event, N to N
Semaphore	Synchronisation with counter for async signalling, N to N
Port	Synchronisation with exchange of Packet, N to N
FIFO queue	Buffered, async communication, except on FIFO full or empty, N to N
Resource	Logical resource to guard critical section (with priority inheritance)
Memory pool	Linked list of memory blocks

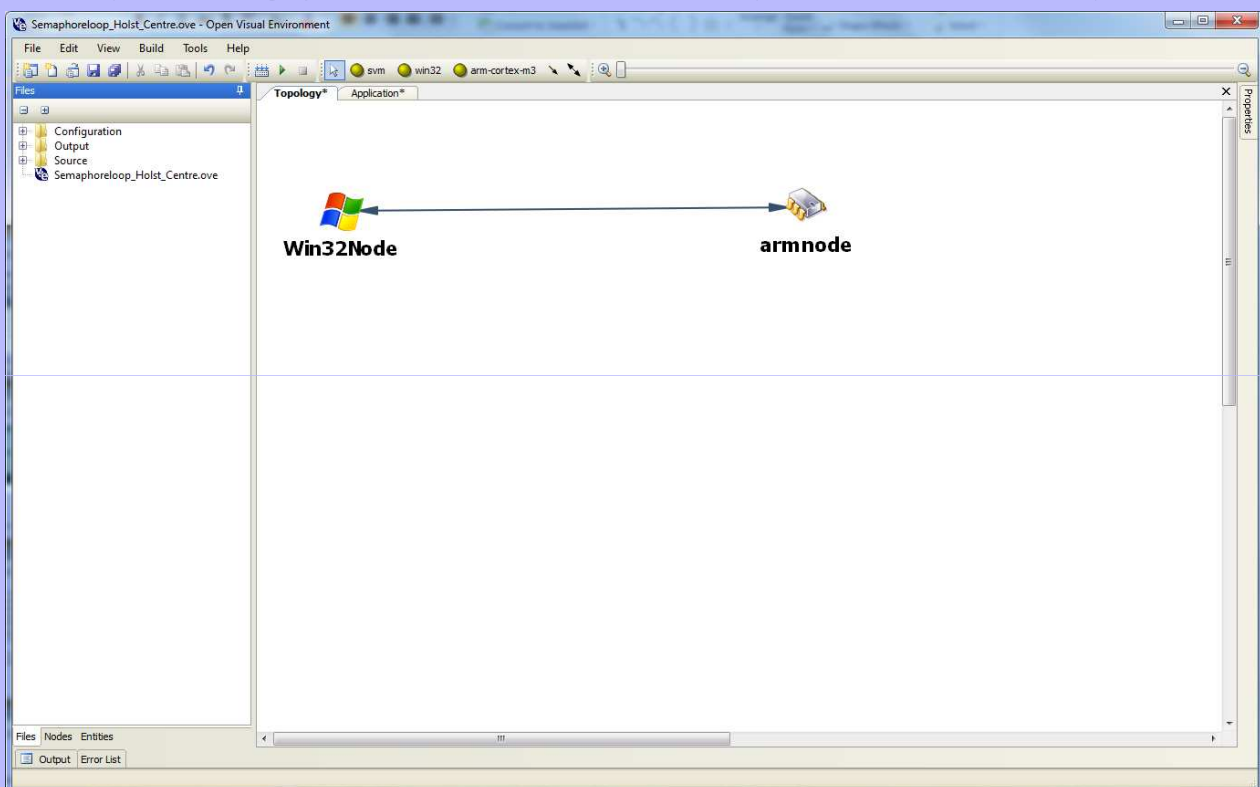
Synchronisation	Semantics
_NW	Non waiting => returns immediately
_W	Waiting until synchronisation (blocking)
_WT	Waiting with a TimeOut.

Interacting Entities Tooling

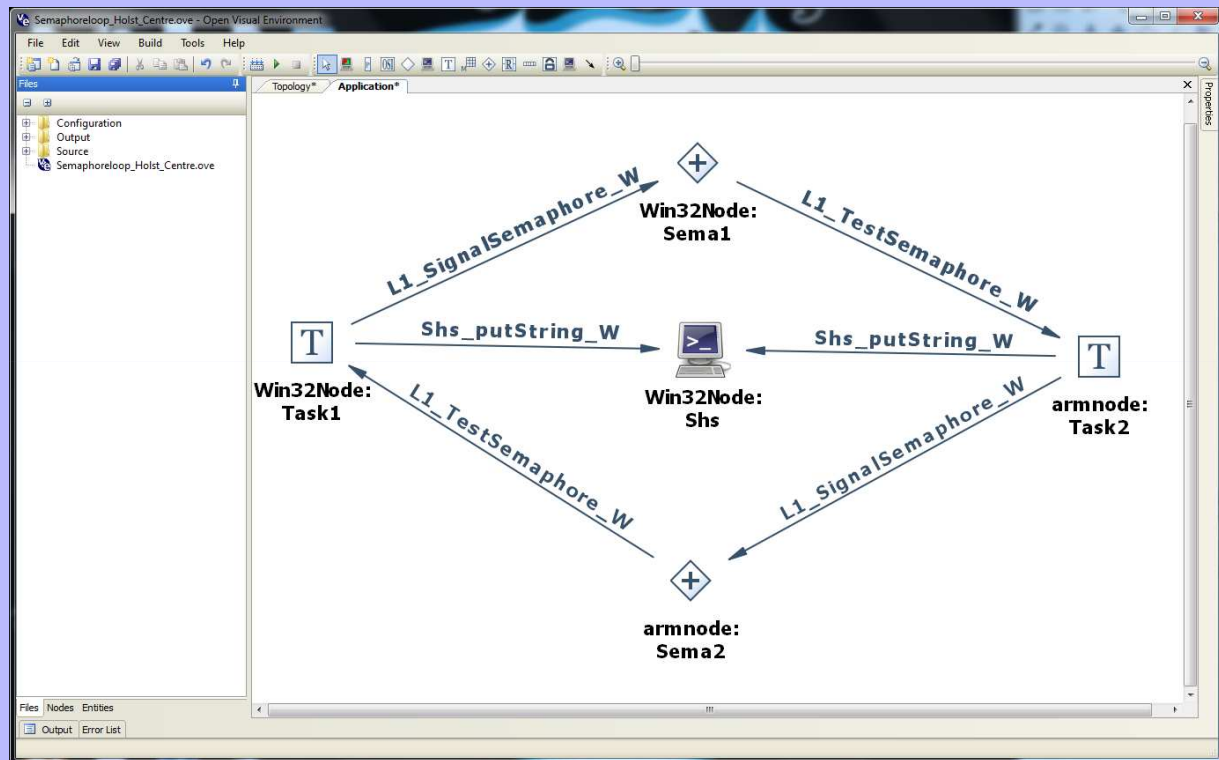


- OpenComRTOS Designer, for Modeling:
 - Topology Diagram, defines the Hardware Setup
 - Application Diagram, defines the Entities and their Interactions
 - Source Code, defines the sequential parts.
- Code Generators: Generate the configuration for OpenComRTOS from the Models
- OpenComRTOS: Runtime Layer providing the Interacting Entities.

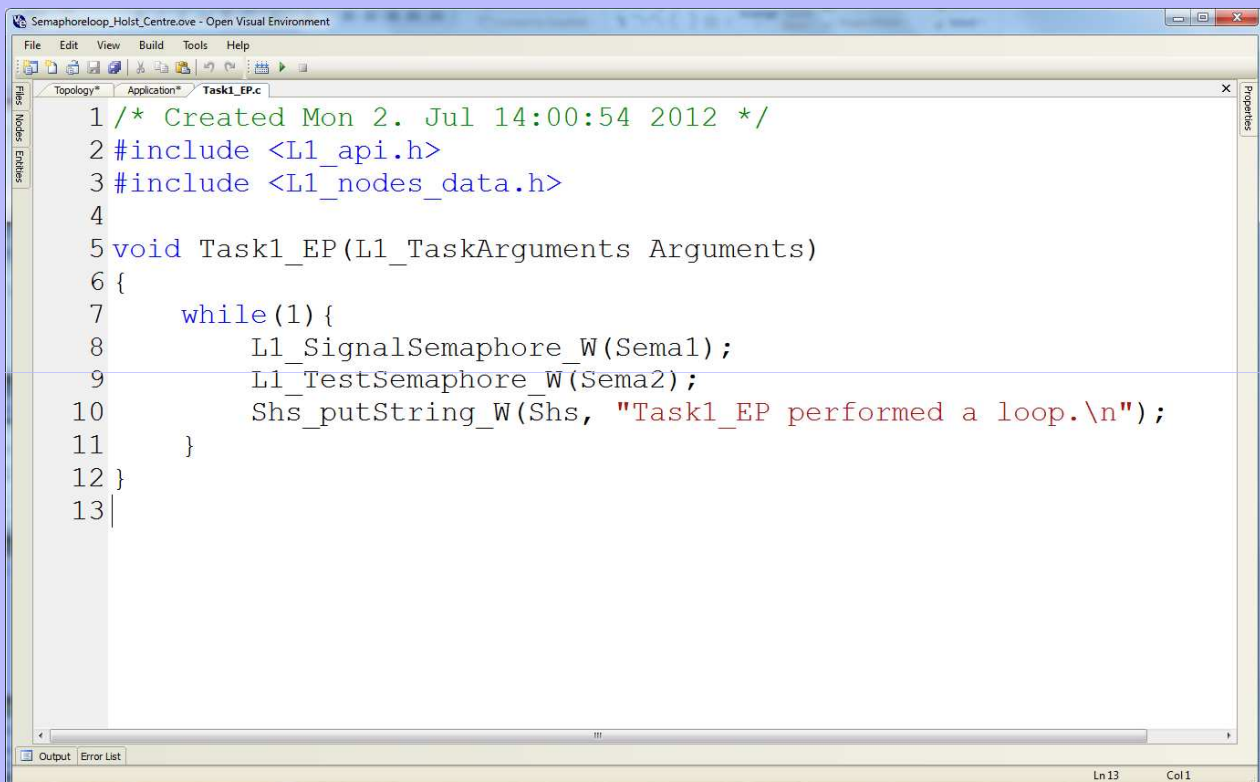
Topology of the Semaphore Loop



Application Diagram in OpenVE



Source Code for Task1



```
1 /* Created Mon 2. Jul 14:00:54 2012 */
2 #include <L1_api.h>
3 #include <L1_nodes_data.h>
4
5 void Task1_EP(L1_TaskArguments Arguments)
6 {
7     while(1) {
8         L1_SignalSemaphore_W(Sema1);
9         L1_TestSemaphore_W(Sema2);
10        Shs_putString_W(Shs, "Task1_EP performed a loop.\n");
11    }
12 }
13 |
```

Codesize, yes it still matters



- Up to 10x smaller than traditional design (thanks to formal development)
- Less power, less memory, easier to verify, scalable ...

CPU Type	Codesize
ARM-Cortex-M3	2.5 – 4.0 kB
XMOS-XS1	5.0 – 7.5 kB
PowerPC e600	7.1 – 9.8 kB
TI-C6678x (8 core DSP)	5.1 – 7.7 kB
Intel-SCC (48 Pentium cores)	4.3 – 5 kB

Code size figures (in Bytes) obtained for our different ports, -Os

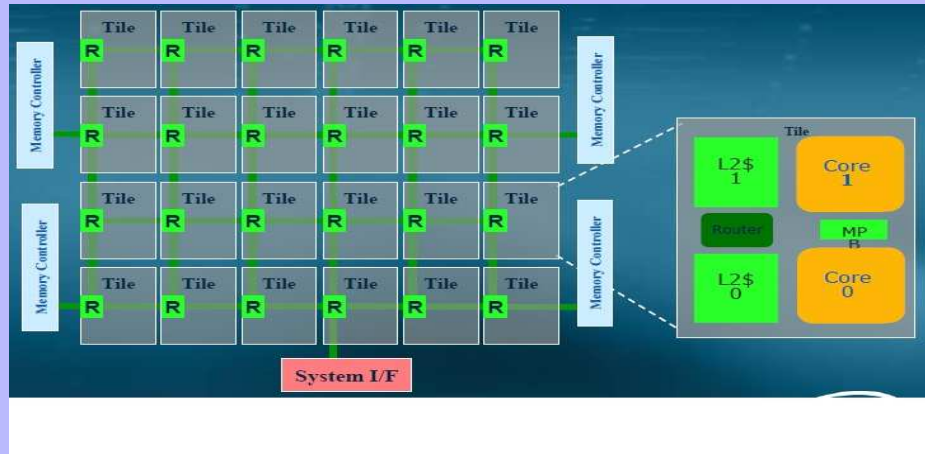
Dormant ports: MLX16 (2K), Xilinx MB (5K), Leon3 (5K), CoolFlux DSP (2K)

Measurement Results



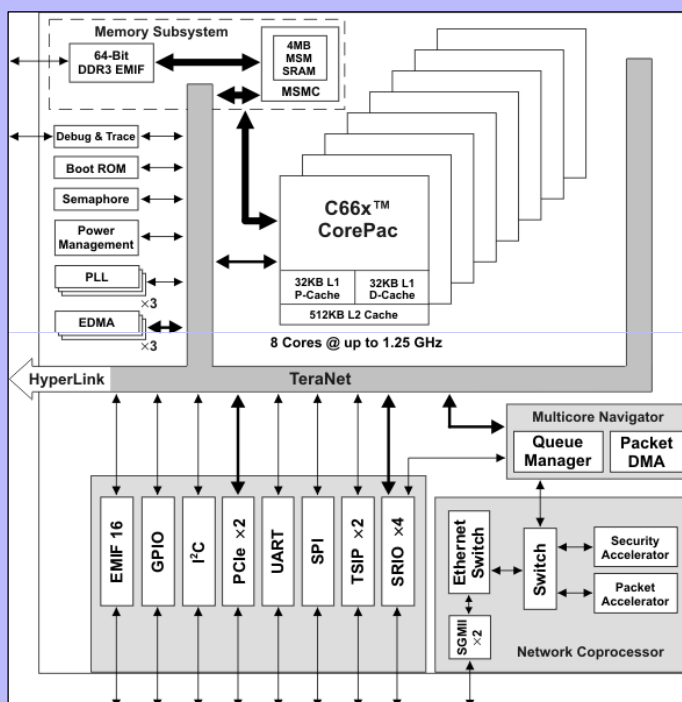
Comparing Intel 48-core SCC and TI 8-core TMS320C6678

OpenComRTOS on Intel 48Core SCC



- L1 cache: 16 KB
- L2 cache: 256 KB
- RTOS kernel on each core
- Communication using memory of MPB (16 KB)

OpenComRTOS on TI TMS320C6678



- “RoC” (Rack On a chip)
- Hierarchical Interrupt Routing up to 1000 possible interrupts per core!
- One Kernel Instance per core
- Program and data in L2 (SRAM) cache
- No DMA for these tests
- DMA added later

Interrupt Latency Measurements



- Two Types of Latencies:
 - IRQ to ISR
 - IRQ to Task
- Measured by using an automatic reload counter as Interrupt Source (IRQ).
- Application Diagram:



Interrupt latencies



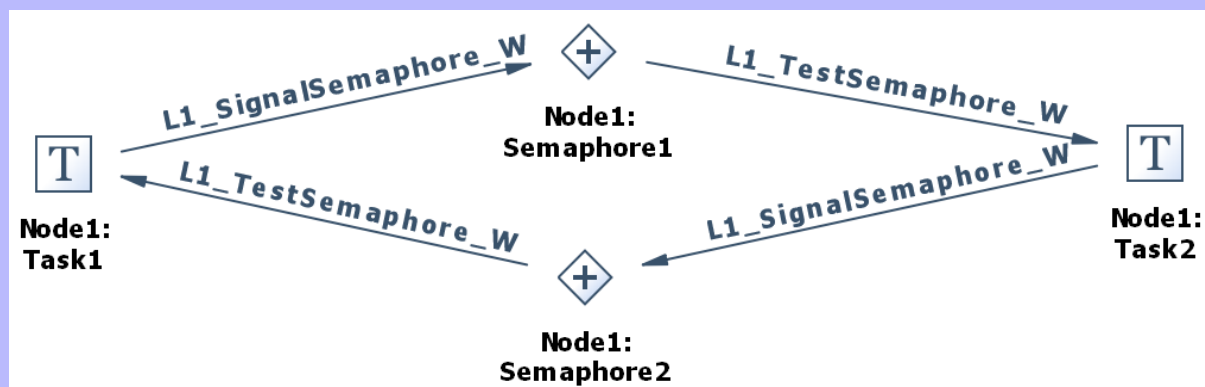
	Intel-SCC (533 MHz)	TI-C6678 (1 GHz)	ARM-M3 (50 MHz)
IRQ to ISR	349 cycles	136 cycles	15 cycles
IRQ to Task	5501 cycles	1367 cycles	600 cycles
Maximum Interrupts per second to ISR	1,527,221	7,352,941	3,333,333
Maximum Interrupts per second to Task	96,891	731,529	83,333

ARM-Cortex-M3 used as reference

Code sizes (8bit per Byte)

	MLX16	uBlaze	Leon3	ARM-M3	XMOS	TI-C6678	Intel-SCC
L1_Hub	400	4756	4904	2192	4854	5104	4321
L1_Port	4	8	8	4	4	8	7
L1_Event	70	88	72	36	54	92	55
L1_Semaphore	54	92	96	40	64	84	64
L1_Resource	104	96	76	40	50	144	121
L1_FIFO	232	356	332	140	222	300	191
L1_PacketPool	--	296	268	120	166	176	194
All L1 services	1048	5692	5756	2572	5414	5908	4953

Semaphore loop test (SP)



- Tasks and semaphore on same node
- Good measure of kernel overhead
- One loop
= 4 context switches + 4 service requests

Semaphore loop times (SP)



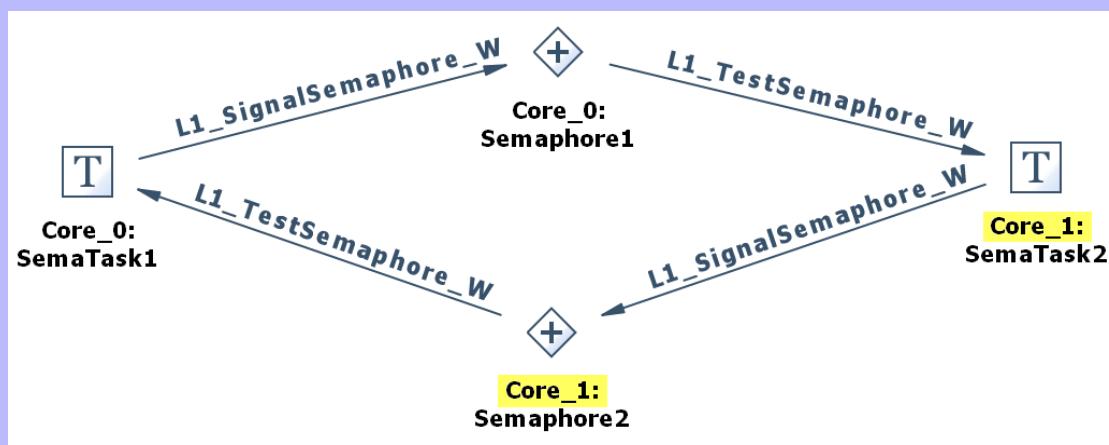
	Clock Freq	Context Size	Memory location	Loop time microsecs	Loop Time cycles
ARM M3	50 MHz	16x32	internal	52.5	2625
NXP CoolFlux	--	70x24	Internal	--	3826
XMOS	100 MHz	14x32	Internal	26.8	2680
Leon3	40 MHz	32x32	Internal	136.1	5444
MLX-16	6 MHz	4x16	Internal	100.8	605
MicroBlaze	100 MHz	32x32	internal	33.6	3360
TI-C6678	1000 MHz	15x32	L2-SRAM	4.5	4500
Intel SCC	533 MHz	11x32	external	4.9	2612

Inter Core Communication on SCC



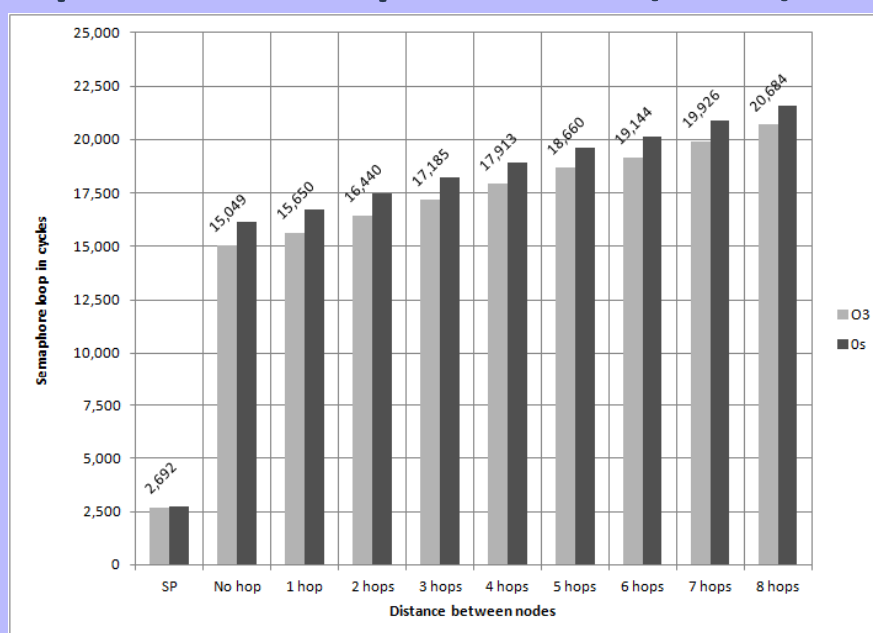
- MPB treated as a FIFO, any Core can write to them.
- Atomic Variables used as locks.
- Operation:
 1. Acquire the Lock for the MPB to write to;
 2. Write the Message to the MPB;
 3. Release the Lock;
 4. Trigger LINT0 on the receiving Core.

Semaphore loop test (MP)



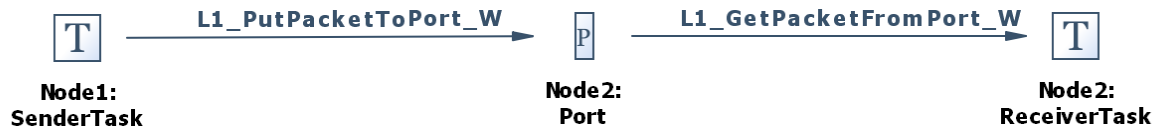
- Tasks and semaphore on different nodes
- Good measure of overhead of kernel + drivers + communication latency

Semaphore loop times (MP)



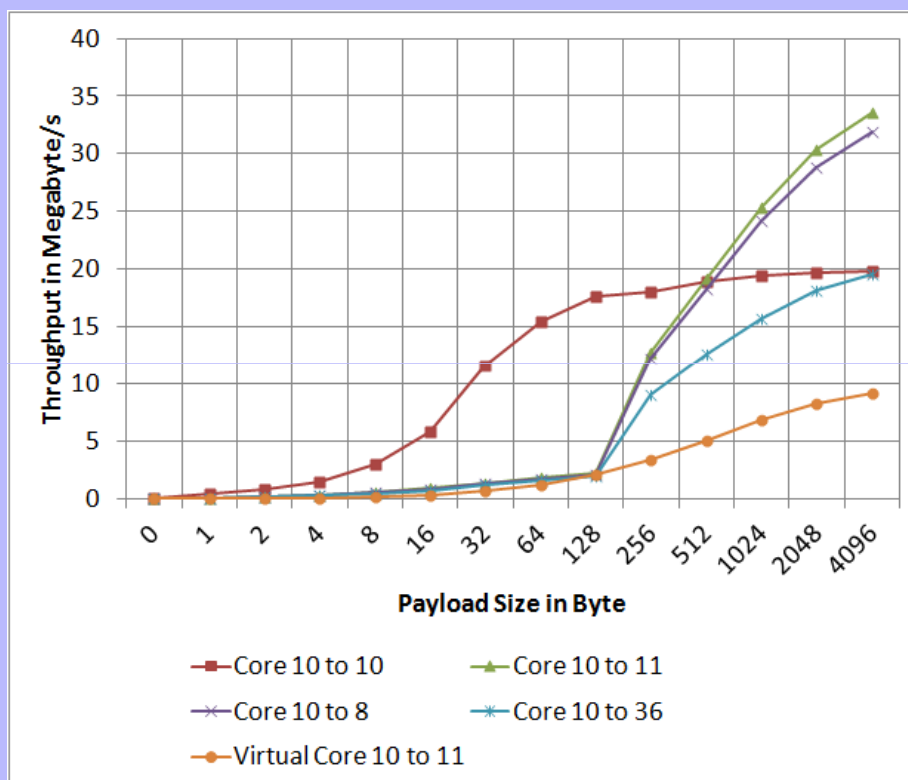
- Extra delay due to TX and RX drivers + communication latency over MPB memory

Communication throughput

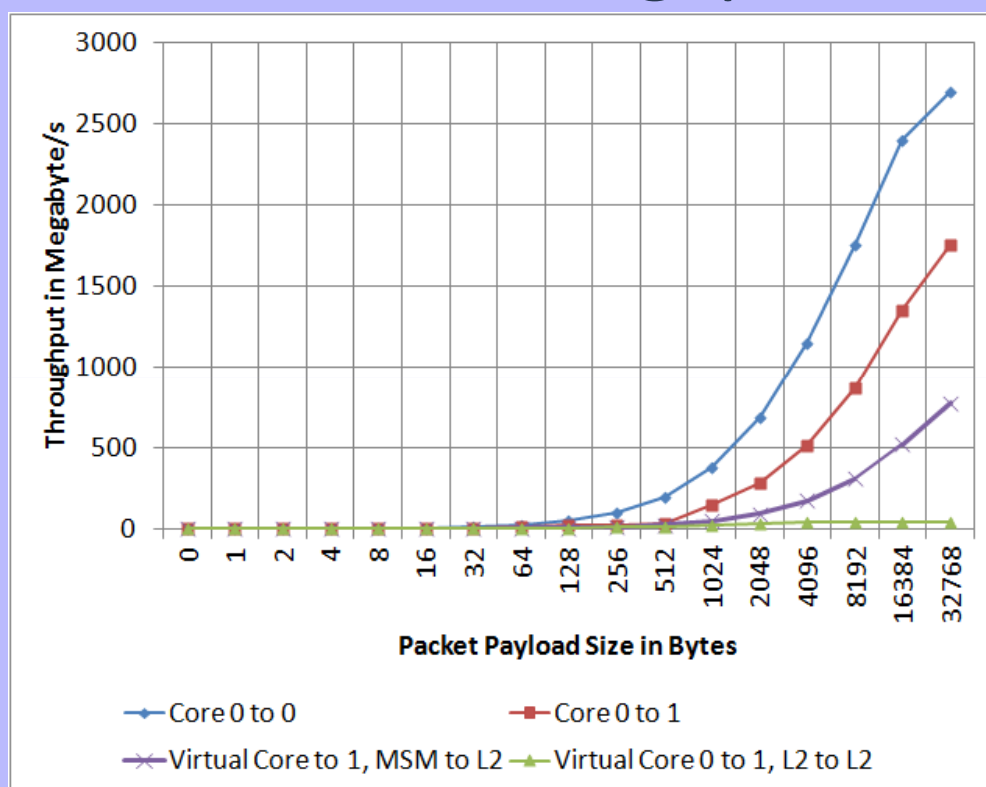


- Uses OpenComRTOS packet switching
- Task synchronise first in Port Hub (=> ACK)
- Data copied from send packet to receive packet in Port Hub

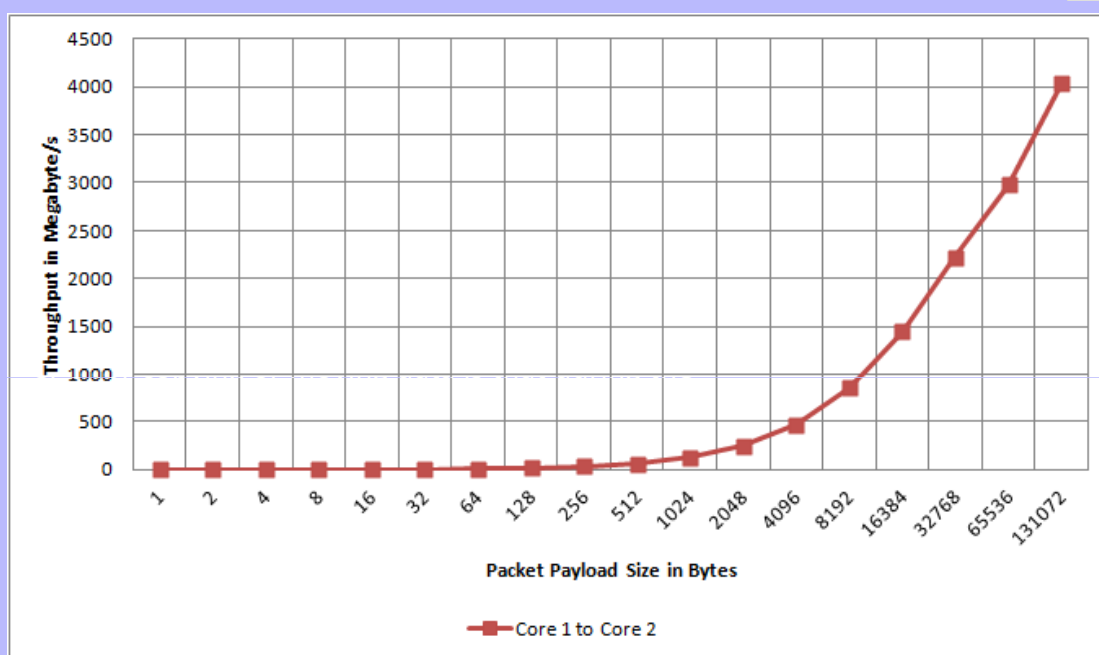
Communication Throughput SCC



Communication Throughput TI



EDMA3 Throughput on TI-C6678



Uses a Device Driver Hub to interface to the EDMA3 Unit (new concept).

Conclusions



- Getting best and predictable real-time performance on modern multi-core is complex:
 - Documentation barrier + hardware complexity
 - Wait states => latencies
 - Cache flushing adds extra overhead
 - Node and memory mapping dependent performance
 - Shared busses = shared resources = extra latency
- Best options: Keep It Simple and Smart:
 - Nice to have features cost memory and cycles
 - Shared resources to be avoided
 - Best is point-to-point + DMA

Further Work



- Integration of the Intel-SCC port into OpenComRTOS Designer:
 - Code generators and platform meta-models
 - Device Drivers
- Optimizations of the TI-C6678 Port
 - Further simplifications of the EDMA3 device driver;
 - Code generators;
 - SoC and Board support packages
- QoS based resource scheduling (Artemis CRAFTERS project)

Contact:



www.altreonic.com

- bernhard.sputh (@) altreonic.com
- eric.verhulst (@) altreonic.com

- Thanks for your attention