OCEOSmp - a multicore RISC-V RTOS to police high-reliability applications

RISC-V in Space Workshop

14th December 2022

Michael Ryan, CTO, O.C.E.Technology Ltd

The police ?

- **Things can go wrong ..**
  - **Best to anticipate crime before it actually occurs**
  - **Keep an eye on the likely culprits**
- **All designs are based on assumptions**
  - **Are things happening more often than expected?**
  - **Are things taking longer than expected?**
  - **Are things taking more space than expected?**
  - **Are expected things not happening (or not being noticed)?**
  - **Is the environment worse than expected?**
- **Is hardware developing a fault?**
  - **Do harts agree?**
- **Robust societies require policing…**

OCEOSmp on Polarfire

System policing vs. application policing

- **System police automatically collect data on likely culprits**
  - **Respond automatically in some cases**
    - **Update system log**
    - **Call application function**
    - **Exit RTOS if needed**
- **Application police can check system data and own data**
  - **Policing checks (low priority task?)**
  - **Policing tests (higher priority task?)**
- **Application police can use system police to handle miscreants**
  - **Take hart/s out of use**
  - **Release hart/s back into use**
  - **Disable tasks, enable tasks**
  - **Kill tasks that are in execution**
  - **Exit RTOS with appropriate code**

OCEOSmp on NOEL-V

OCEOSmp: design (1)

- **Support policing**
  - **Static allocation of activity records**
  - **System log used by system and application**
  - **Record and log still visible if have to exit**
- **Exclude some temptations - stack resource policy (Baker 1991)**
  - **Unbounded priority inversion cannot occur**
  - **Chained blocking cannot occur**
  - **Deadlocks excluded if single hart, warning otherwise**
- **Save space – stack resource policy (Baker 1991)**
  - **Single system stack per hart rather than per task**
- **Symmetric – all harts used equal after initialisation**
  - **Harts can be reserved for other uses, e.g. Linux**
  - **Harts can be reserved for higher priority tasks**
- **Support schedulability analysis**

OCEOSmp on NASA HPSC

OCEOSmp: design (2)

- **Task priority fixed**
  - **based on task importance**
- **Task pre-emption threshold**
  - **task pre-empted only by tasks with higher priority than threshold**
- **Multiple execution instances**
  - **multiple same task ‘jobs’ can be in execution typically using different data**
- **Same priority jobs run to completion in turn**
  - **no time slicing**
- **Timed actions independent of scheduling**
  - **being early can also be a problem…**
    - **data output at specific time**
    - **task start request at specific time**

OCEOSmp for SiFive

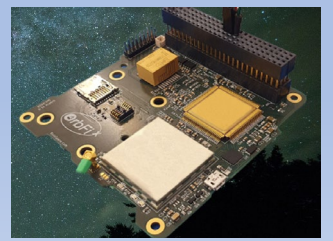
OCEOSmp: design (3)

- **Mutexes**
  - **unbounded priority inversion cannot occur**
  - **deadlock warning, cannot occur if single core**
- **Read-Write Mutexes**
  - **allow simultaneous reads when not held for write**
  - **prevent writing if being read**
  - **prioritize write requests over reads**
  - **order writes first come first served**
  - **unbounded priority inversion cannot occur**
- **Counting Semaphores**
  - **support wait with timeout**
- **Data Queues**
  - **support read with timeout**

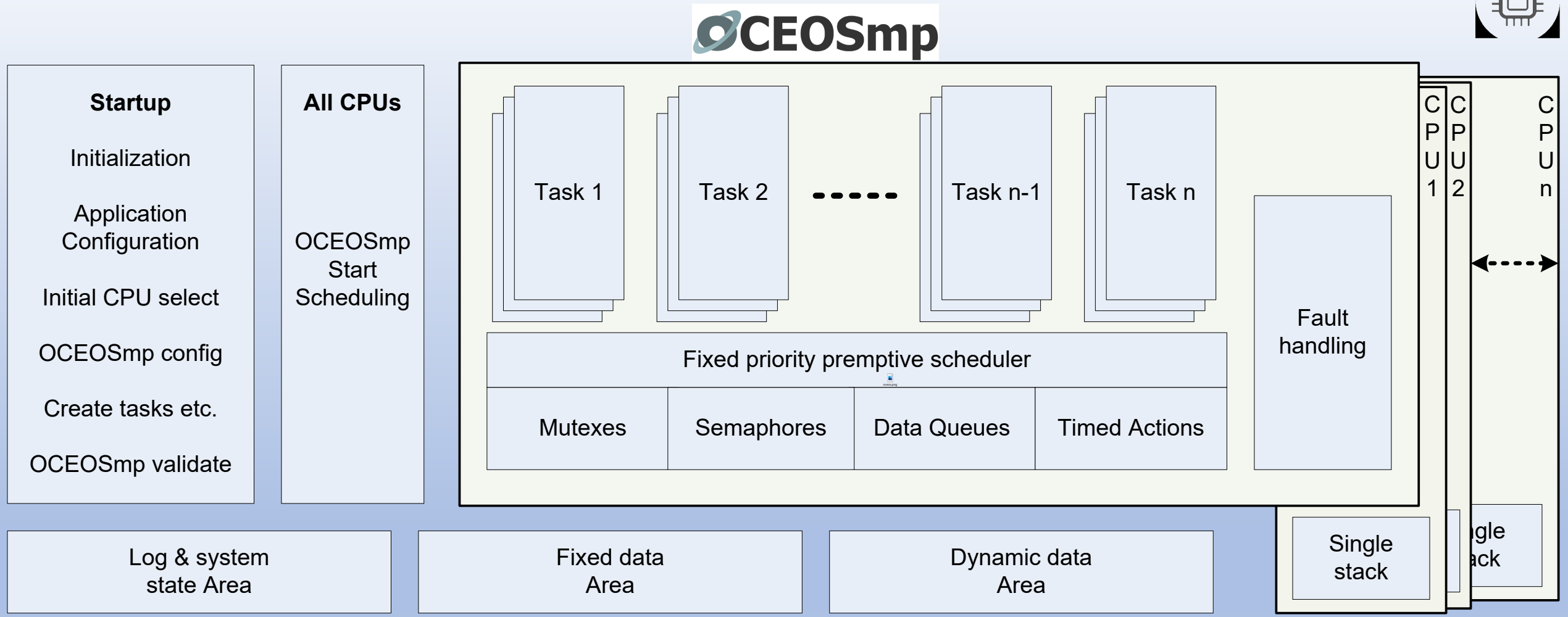
Solar array drive

OCEOSmp: design (4)

- **System time**
  - **in microseconds, 64 bit**
- **Context switch timing**
  - **context switching shared across all cores**
  - **context switch time minimized**
- **Interrupts**
  - **interrupt disabled timing minimized**
  - **high priority timer interrupt reserved for timed actions**
- **Some numbers**
  - **1 <= cores <= 255**
  - **1 <= tasks <= 255, 1 <= execution instances (jobs) <= 255*15**
  - **0 <= mutexes <= 63, 0 <= 63 read-write mutexes <= 63**
  - **0 <= counting semaphores <= 63, 0 <= data queues <= 63**
  - **memory < 20 KiB**



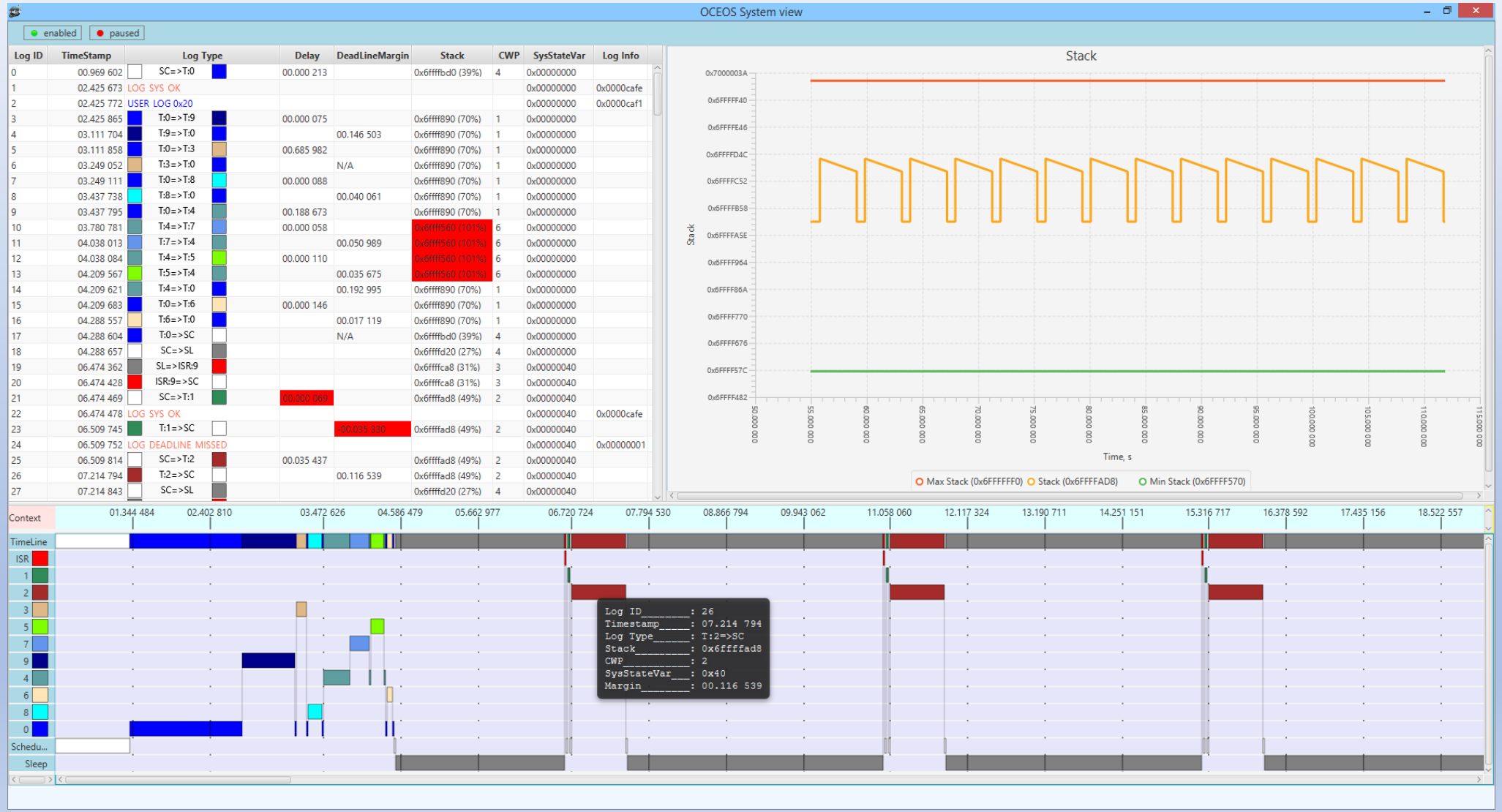
GNSS system using OCEOS


CEOSmp
Startup
Initialization
Application Configuration
Initial CPU select
OCEOSmp config
Create tasks etc.
OCEOSmp validate
All CPUs
OCEOSmp Start Scheduling
Task 1
Task 2
Task n-1
Task n
Fixed priority premptive scheduler
Mutexes
Semaphores
Data Queues
Timed Actions
Fault handling
CPU 1
CPU 2
CPU n
Single stack
Log & system state Area
Fixed data Area
Dynamic data Area

OCEOSmp: Using it

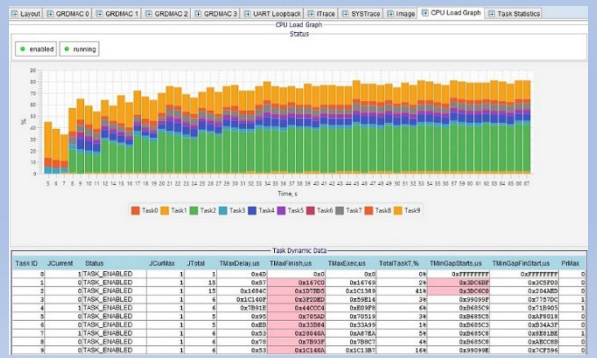
- **Library** - components not used not linked into the executable
- **Servant not Master** – started by application main()
- **Step 1 : Create application configuration, pass to *oceos_init()***
  what cores to use, what stack space, log entries
  how many tasks, jobs per task, timed actions,
  how many mutexes, semaphores, data queues
- **Step 2: Create corresponding tasks, mutexes, etc.**
  using *oceos_task_create()* etc.
- **Step 3: Use oceos_init_finish() to complete fixed data and checksum**
- **Step 4: Pass fixed data and initial task (if any) to *oceos_start()***
  dynamic data area is set up
  multi-core scheduling begins

Debug support - DMON



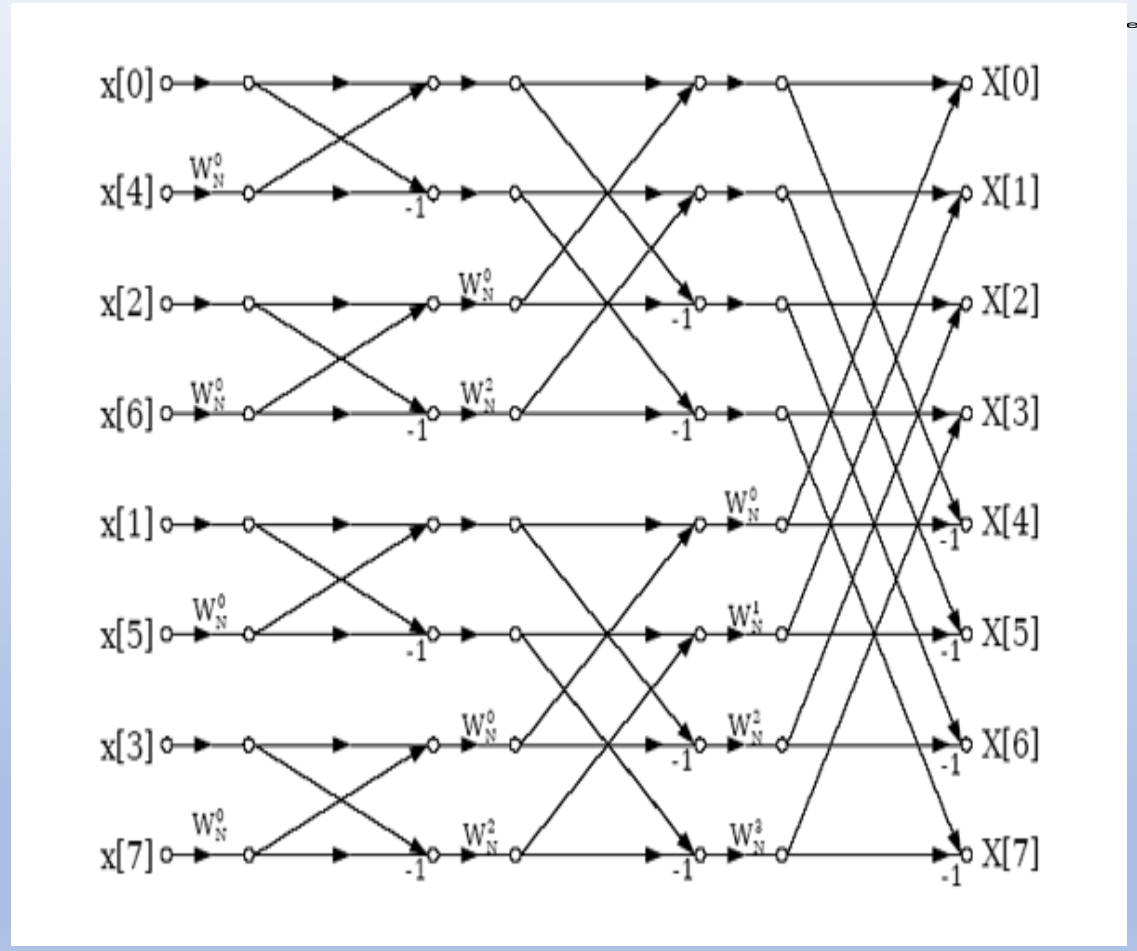
Current Status

- **OCEOS (single core)**
  - **SPARC and ARM versions complete (with additional support for GR716 microcontroller) ESA Flight Level B qualification ready**
- **OCEOSmp (multicore)**
  - **Checking initial scheduler design on multicore SPARC & RISC-V (ARM later)**
  - **Initial quad core results (Gaisler GR740 and Microchip PolarFire RISC)**
    - **Check scheduler distributes work symmetrically**
      - **1001 task starts : Per CPU 251,250,250,250**
    - **Check task can be run in parallel to provide speed up**
      - **4096 sample FFT (complex number entries)**
      - **one task, four jobs in parallel on different parts of FFT**
      - **speedup factor 3.6 - 3.7 (haven't tried to optimise)**
  - **Design closure pending with ESA**
- **Availability**
  - **OCEOS - single-core development kit on-sale**
  - **OCEOS - multicore beta evaluations available soon**



OCEOS task usage & debug screen

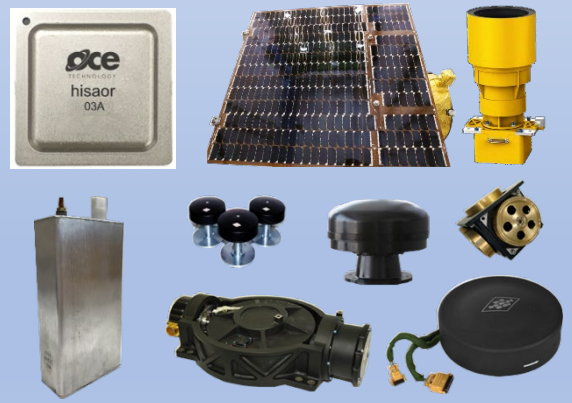
FFT



CPU0	CPU1	CPU2	CPU3	% Processing (4096 array)
Initialisation Start CPUs				**Preprocessing**
	Startup Power down	Startup Power down	Startup Power down	
Setup control block Restart CPUs				
¼ Rearrange	¼ Rearrange	¼ Rearrange	¼ Rearrange	(rearrange if necessary)
¼ Main FFT n-2 stages where array size = 2^n	¼ Main FFT n-2 stages where array size = 2^n Power down	¼ Main FFT n-2 stages where array size = 2^n Power down	¼ Main FFT n-2 stages where array size = 2^n Power down	**Main Stage** 84%
Setup control blocks for 2nd last stage Restart CPUs				
Process ¼ 2nd last stage	Process ¼ 2nd last stage Power down	Process ¼ 2nd last stage Power down	Process ¼ 2nd last stage Power down	**2nd Last** 5%
Setup control blocks for last stage Restart CPUs				
Process ¼ last stage	Process ¼ last stage Power down	Process ¼ last stage Power down	Process ¼ last stage Power down	**Last Stage** 5%
Setup control blocks Restart CPUs				
Process ¼ conversion	Process ¼ conversion	Process ¼ conversion	Process ¼ conversion	**Conversion Stage (if necessary)**

Finally

- **Thanks to ESA for their support**
- **Thank you for listening**
- **Any Questions?**
- **Any Answers?**
  - **Would welcome ideas or suggestions e.g. policing data that might be of interest**



michael.ryan@ocetechnology.com

www.ocetechnology.com