



SITAEL Contribution to ESA IP Cores

2nd IP Core Presentation Day
Noordwijk (NL), September 16th, 2013

OUTLINE

- ❑ **CCIPC: CANopen Controller IP Core**
- ❑ **IMMIPC: Improved Memory Module IP Core**
- ❑ **V8uC: SPARC V8 Micro-Controller IP (V8uC)**
- ❑ **Final Conclusions**

CANOpen Controller IP Core (CCIPC)

CANopen Controller IP Core (CCIPC)

❑ **CCIPC was developed in the frame of ESA ExoMars Programme, under a contract with TAS-I**

❑ **CCIPC functions:**

- CANOPEN Slave Node
- Object Dictionary
- PDO (TPDOs and RPDOs) objects
- SDO (expedited, segmented and Block) service
- Network Management state machine with the relative services
- SYNC consumer
- Heartbeat producer and consumer
- Redundancy Manager

❑ **CCIPC Configuration capability:**

- System Frequency : 10 – 16 MHz
- CAN Bit Rate supported: 125Kbit/s, 250Kbit/s, 500Kbit/s, 1Mbit/s
- Node-ID & Node Count
- Three selectable Host Interfaces:
 - Generic I/O interface
 - AMBA interface
 - Direct Interface
- SDO handler (not Mandatory)
 - expedited (not Mandatory if the PDO service is configured)
 - expedited + segmented
 - expedited + block

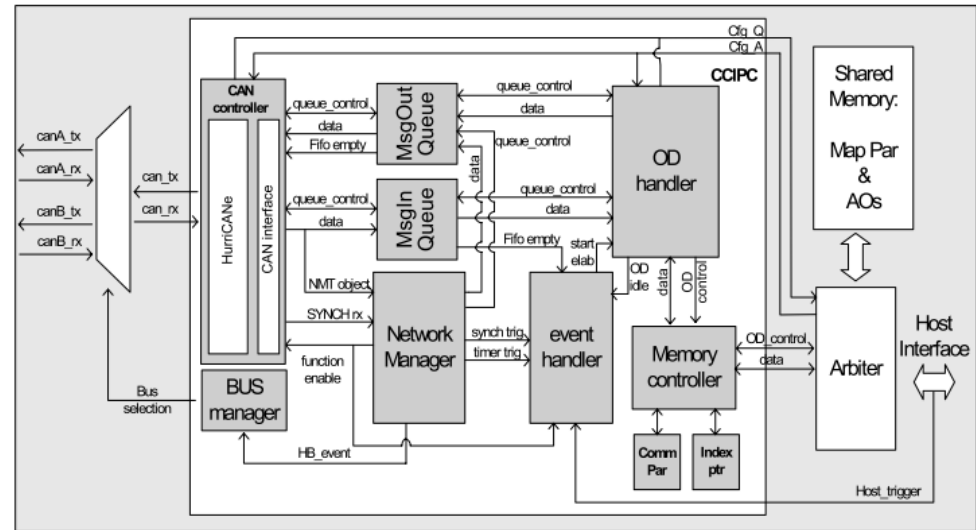
❑ **Target technology requirement:**

- The target FPGA technology for the verification is the Microsemi A3PE3000
- The target FPGA technology is the Microsemi RTAX FPGA family

CCIPC Architecture

The CCIPC Building Blocks are:

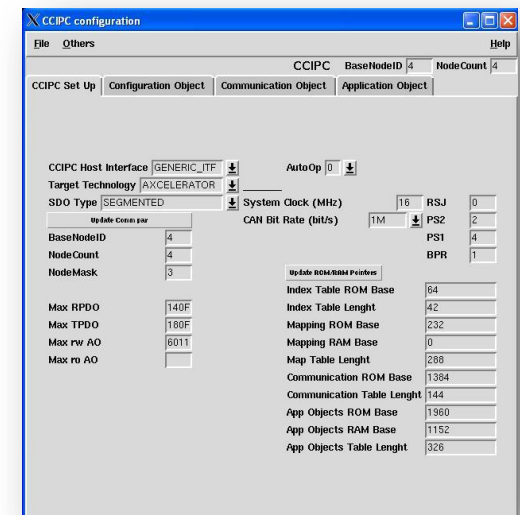
- ☐ BUS manager
- ☐ CAN controller
- ☐ MsgIn/MsgOut Queue
- ☐ Event Handler
- ☐ OD Handler
- ☐ Network Manager
- ☐ Memory Controller
- ☐ Communication Parameter
- ☐ Index Pointer



Registers	Description
HurriCANE Configuration	HurriCANE reset & CAN Bit Rate
HurriCANE & CCIPC status	Status of: <ul style="list-style-type: none"> HurriCANE (Rx/TX error, BusOff & Error Passive) CCIPC (NMT state, Active Bus)
IRQ handler	Manager of CCIPC IRQ conditions: <ul style="list-style-type: none"> PDO completed/error SDO completed/error Synch elaboration completed/error (SWL expiration) Bus switch
EDAC Controller	EDAC error manager. Two flags are used to enable/disable EDAC on : <ul style="list-style-type: none"> CCIPC memory modules (External & Internal) CCIPC queues

CCIPC Configuration Tool

- ❑ Each CANOpen node instance has to be configured by defining the needed services and the Object Dictionary
- ❑ The CCIPC provides a GUI to simplify the generation of the user-defined CCIPC configuration like:
 - CANOPEN services
 - CCIPC interface
 - CAN bit-rate
- ❑ The CCIPC GUI can read, edit and write the CANOPEN standard DCF or EDS to build the user-defined object dictionary:
 - Configure RPDO/TPDO
 - Configure Synch consumer
 - Configure COB-IDs
 - Configure Heartbeat consumer/producer time
 - Configure redundancy management parameters
- ❑ **CCIPC GUI OUTPUT:**
 - MY-conf.dcf
 - My-conf.vhd (containing the VHDL configuration parameters)
 - My-OD.txt (containing the ROM content, EDAC included)



CCIPC Synthesis Result

The following configuration has been used with Generic I/O & AMBA Interface:

- ☐ 8 Node-ID supported (Mask size = 3);
- ☐ 32 RPDO (12 synchronous, 14 asynchronous, 6 disabled);
- ☐ 32 TPDO (10 synchronous, 16 asynchronous, 6 disabled);
- ☐ SDO server (SDO block & Expedited supported)
- ☐ 30 Manufacturer Specific Application Objects

The following configuration has been used with Direct Interface:

- ☐ 1 Node-ID supported (Mask size=0);
- ☐ 1 Asynchronous RPDO & 1 Asynchronous TPDO;
- ☐ SDO Expedited server;
- ☐ 1 Read-Write Manufacturer Specific entry

CCIPC Synthesis Result

- ❑ The Microsemi RTAX1000 FPGA is the target device used for the synthesis process of the CCIPC module and the operating frequency is above 20 MHz @100krad
- ❑ Max 55% of RTAX1000 resource utilization
- ❑ 15% of the CCIPC is dedicated to HurriCANE implementation

Direct

Module	Total cell	RAM (512x9)	Freq. (MHz)
CCIPC	8720	7	24
MapAOTable	339	--	
Other logic	169	--	
TOT	9401	7	

AMBA

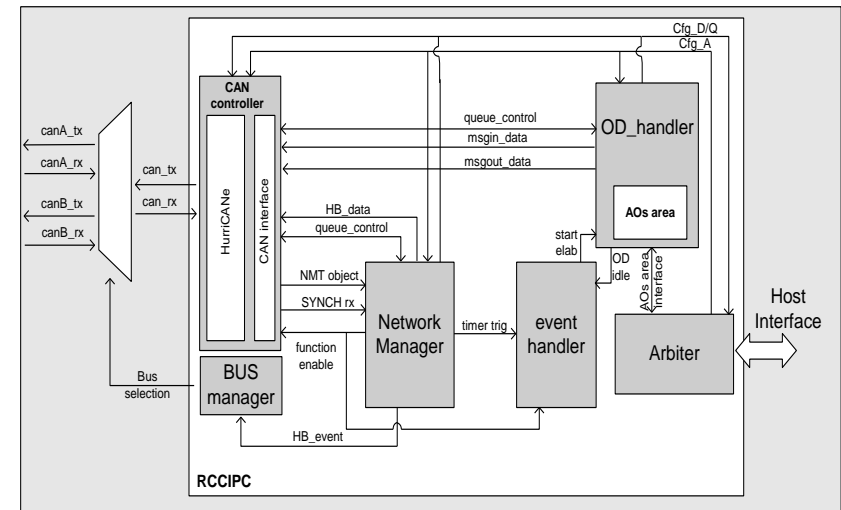
Module	Total cell	RAM (512x9)	Freq. (MHz)
CCIPC	9384	17	22
AHB_master	574	--	
AHB_slave	290	--	
Other logic	108	--	
TOT	10356	17	

Generic I/O

Module	Total cell	RAM (512x9)	Freq. (MHz)
CCIPC	9353	17	23
Arbitrer	507	--	
Other logic	48	--	
TOT	9912	17	

The RCCIPC is a reduced version of the CCIPC with following characteristics:

- ❑ 1 Node-ID supported;
- ❑ NMT state machine with the relative services;
- ❑ SYNC consumer;
- ❑ Heartbeat producer and consumer;
- ❑ Redundancy Manager;
- ❑ 1 Asynchronous RPDO;
- ❑ 1 Asynchronous TPDO;
- ❑ SDO Block server;
- ❑ 3 Manufacturer Specific entries:
 - 1 Array U32 (2 sub-indexes) for TPDO;
 - 1 Array U32 (8 sub-indexes) for RPDO;
 - 1 entries mapped as array of U32/U16/U8 (up to 255 sub-indexes);



Module	Total cell	RAM (512x9)	Freq. (MHz)
RCCIPC	3496	5	23
Arbiter	165	--	
Other logic	38		
TOT	3699	5	

- ❑ The Microsemi RTAX250 FPGA is the target device used for the synthesis process of the RCCIPC module and the operating frequency is above 20 MHz @100krad
- ❑ 88% of RTAX250 resource utilization
- ❑ 38% of the RCCIPC is dedicated to HurriCANE implementation

CCIPC Validation Steps

The CCIPC Validation has been completed in three steps:

❑ Functional RTL Simulation Campaign

- This test Campaign has been performed by SITAEL and TAS-I on the R/CCIPC VHDL model to verify the correct behavior of all the implemented functions

❑ Hardware Validation Test Campaign

- The R/CCIPC Validation has been performed on the CCIPC Engineering Model by using a dedicated test set-up. This test set-up uses a third party tool validated w.r.t. the CIA CANOPEN conformance test

❑ EXOMARS CCIPC Verification

- A representative R/CCIPC configurations for EXOMARS have been verified on Microsemi AX-250 and AX-1000 FPGA.



CANopen Validation Boards

Conclusions

- ❑ **The CCIPC V3.5 and RCCIPC V2.2 databases have been successfully validated and delivered to ExoMars users**
- ❑ **The IP core has been designed following the ECSS CAN BUS Working Group standardization activities and therefore it will be compliant with the incoming ECSS-E-50-15C**
- ❑ **The R/CCIPC have been verified on Microsemi AX-250 and AX-1000 FPGA**
- ❑ **Possible additional features are (they will be available in the next months):**
 - Define a “special” Node-ID to support CAN Broadcast message
 - Acyclic Synchronous TPDO: PDOs are triggered by a defined device-specific event, but transmitted with reception of the next Sync message

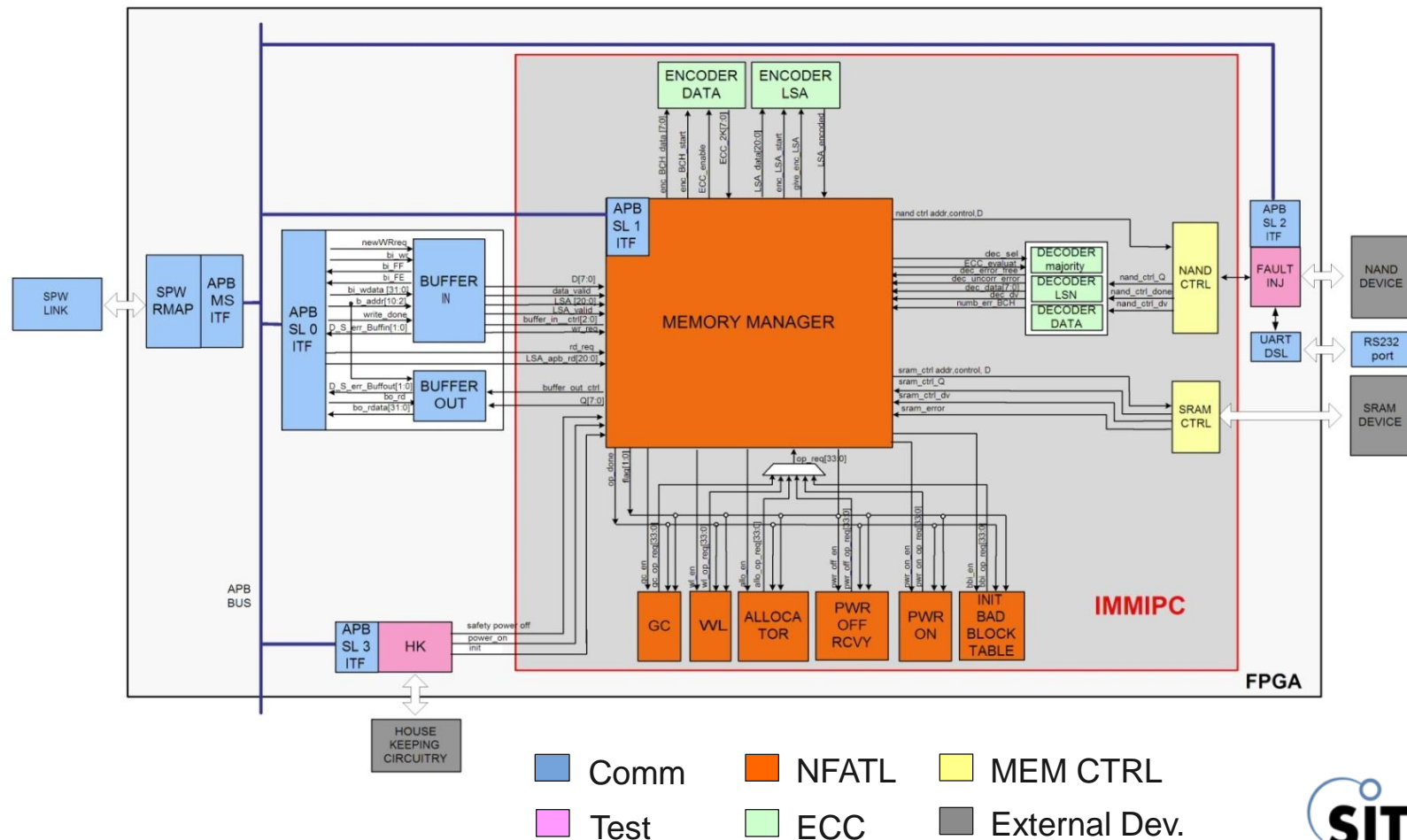
Improved Memory Module IP Core (IMMIPC)

Improved Memory Module IP Core

- ❑ **The Improved Memory Module IP Core (IMMIPC) mitigates the radiation effects caused by heavy ions, neutrons and protons impact on COTS based mass memory modules, making use of digital techniques**
- ❑ **IMMIPC is a module embodying a reliable low-power mass memory module**
- ❑ **Write-Read operations are controlled with a DMA-like interface accessed via SpaceWire-RMAP**

What is in the Black-Box

- ❑ NAND Flash Address Translation Layer (NFATL)
- ❑ Error Correction Code and control for NAND Flash errors and SEU/SEFI effect



NAND Flash Address Translation (1)

❑ **NAND FLASH MEMORIES**

- Flash Memory array is composed of blocks and each block is made up of pages (1 block = 128 pages = 512 Kbyte + Spare)
- Minimum erasable unit is a block
- Erase operation is time consuming (700 us)
- Some blocks can be bad at shipping time
- Each block can be written only a limited number of times (100,000)
- Program page is possible only on erased pages
- Within a block, pages must be programmed in sequential order
- Each page is provided with additional storage called spare area to store metadata (ECC and additional information)



❑ **NAND FLASH ADDRESS TRANSLATION LAYER (NFATL)**

NAND Flash Address Translation (2)

❑ **NAND Flash Address Translation Layer Building Blocks**

- **ALLOCATOR** handles 2K page read and write requests. It manages **overwriting** of the same physical page by means of Logical to Physical address translation with hybrid mapping and log blocks method
- **BAD BLOCKS MANAGER** manages the blocks marked as bad by vendor at shipment time and the blocks that worn out during device life time
- **WEAR LEVELER** evenly distributes the number of erasures of each block to improve device endurance implementing a static wear level method
- **GARBAGE COLLECTOR** reclaims space erasing blocks with obsolete data in case of lack of free pages
- **POWER OFF** saves SRAM tables in dedicated block of NAND Flash for a programmed back-up and before shut down of the NAND Flash
- **POWER ON** reconstructs SRAM tables after a power up retrieving information from NAND Flash dedicated block

❑ **Resource Budget on Microsemi A3PE3000 FPGA :**

- Core cells: 10%
- Block RAMs: 15%

Error Correction Code

All data and metadata in NAND Flash and in off-chip / on-chip RAMs are protected with a specific ECC

The IMMIPC is equipped with 4 distinct codec engines

- ❑ **BCH code protecting 2 KB NAND FLASH data page**
 - Number of parity bits : 225 bits (stored in the spare sector)
 - Random error-correcting capability: 15 bits
 - Codec area occupation on Microsemi ProAsic3 A3PE3000 @ 20MHz: 30% logic resources and 6% Block-RAM
- ❑ **A SEC-DED code protecting the 21 bit Logical Sector Address**
 - Number of parity bits: 7 bits (stored in the spare sector)
- ❑ **A repetition code protecting the binary information of the RAM-table copy in a NAND block**
 - Repetition of '1' or '0' 7 times (in a single byte into spare area)
 - Random error-correcting capability: 3 bits
- ❑ **SEC-DED codes to protect data stored in external SRAM and internal Block-RAMs**

Demo Board Software Suite: API developed in “C” language

❑ Basic functionality :

- Communication: SpaceWire registers and DSU registers write/read accesses
- NAND commands: reset, read, write, block erase, etc
- SRAM: write and read accesses

❑ NFATL verification :

- Allocator: page read, page write and page overwrite
- Wear leveler
- Garbage collector
- Bad blocks identification
- Power-on and power-off

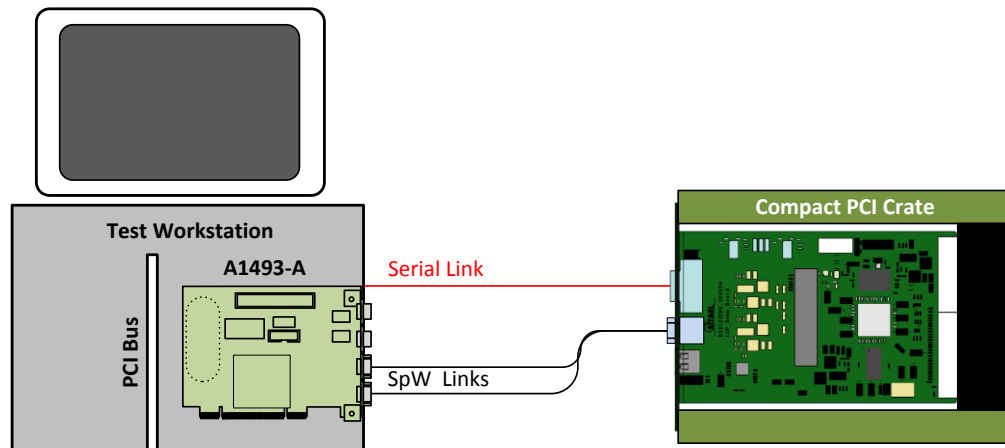
❑ ECC engine: fault injections on decoders (data, LSA, majority byte)

❑ Application Software for files transfer

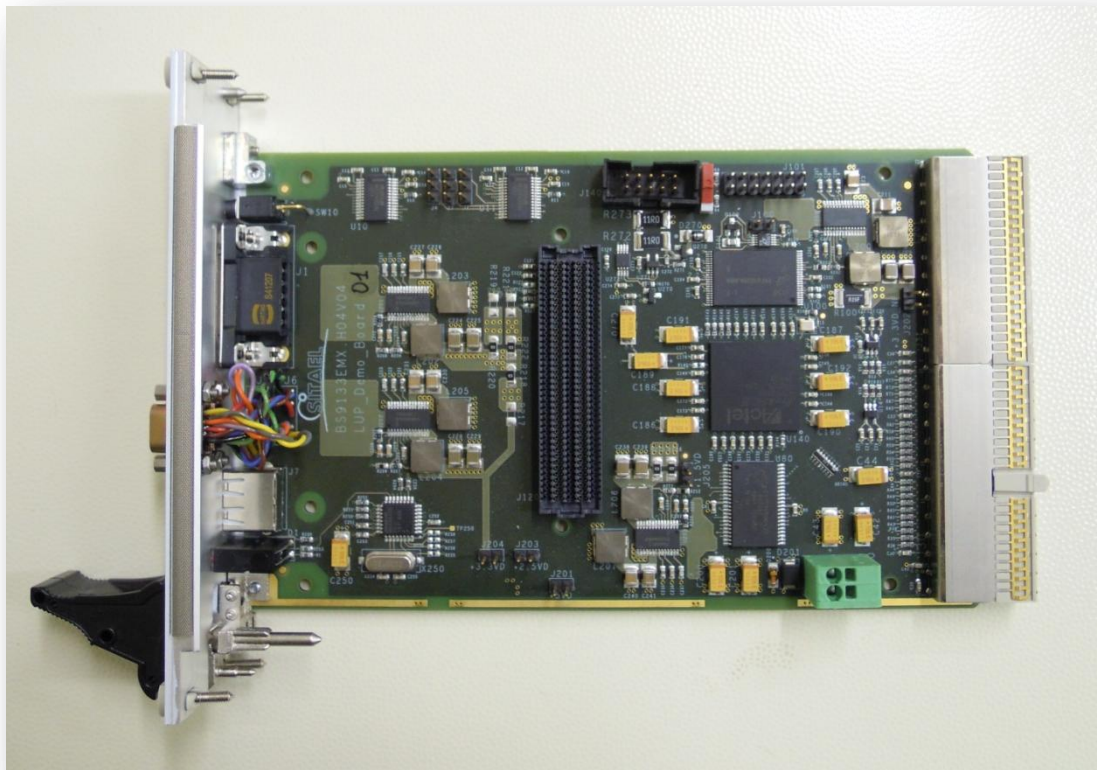
- Files of size up to 280MB have been stored and read into/from NAND Flash
- NAND Flash fulfilled with 8Gb of data

Demo Board (1)

- ❑ CompactPCI 3U-8HP (RASTA compliant)
- ❑ Microsemi ProASIC3E A3PE3000
- ❑ NAND FLASH Micron MT29F64G08AJABA (UUT) 64Gbit with dedicated power Supply
- ❑ SRAM Buffer 1Mx8
- ❑ HK circuit based on I2C current and temperature monitor LTC2990 + Latch-Up emulator
- ❑ SpaceWire 2x ports + Serial RS232



Demo Board (2)



Conclusions

❑ **Project Status**

- IMMIPC architectural and detailed design complete
- Manufacturing completed
- Test on Demo Board completed
- RASTA Environment Integration completed

❑ **Possible additional features**

- Demo Application test case
- NFATL algorithms evaluation/refinement
- Protection of bigger NAND Flash page

SPARC V8 Micro-Controller IP (V8uC)

SPARC V8 Micro-Controller IP (V8uC)

- ❑ **V8uC is an ESA project to realize a 32 bit micro-controller starting from LEON2-FT core data-base**
- ❑ **From CPU to micro-controller a new simplified memory data path had to be designed**
- ❑ **Design approach of reuse of large parts of the LEON2-FT core:**
 - To reduce design timing and risk
 - To inherit its SW tools and libraries
- ❑ **Changes minimized to not reintroduce errors in debugged components**

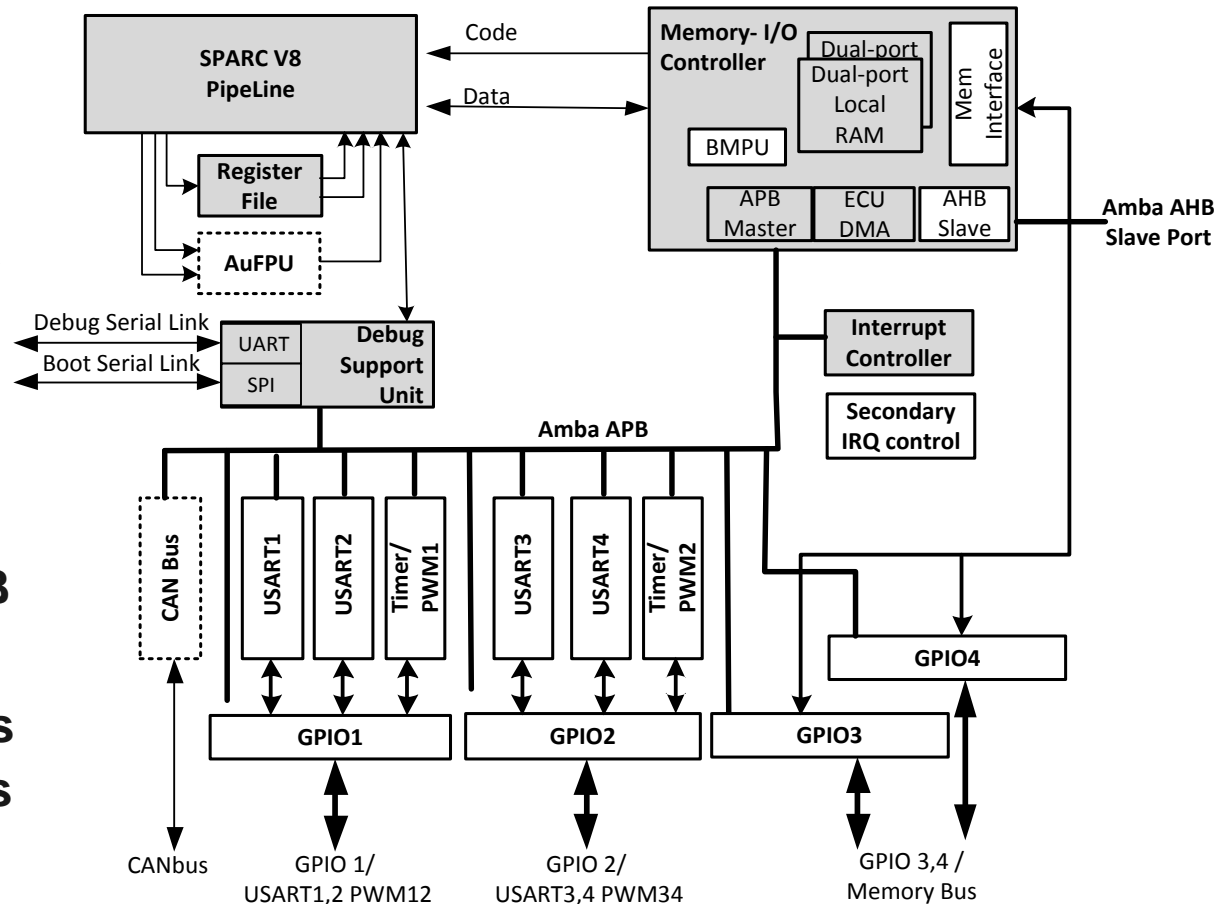
Target Application

- ❑ **Embedded sub-systems where LEON features/costs exceed, but not cover effective requirements**
 - Closed control loop applications
 - Acquisition and serialization of multiple discrete or analogue signals
 - In-line data elaboration
- ❑ **Small Software: no real Operative System but only scheduler loop**
- ❑ **Memory data path simplified with on chip local memory (64/128 Kbyte) sufficient to keep entire application**
- ❑ **Remote boot via serial link as option to save non volatile memory on board**
- ❑ **Several I/O lines and serial peripheral interfaces**

The objective is to cover I/O or control boards requirements without need of additional FPGA design

V8uC Architecture

- ❑ No Caches
- ❑ New Memory Controller with DMA engine
- ❑ Programmable number of USARTs, GPIOs and PWMs
- ❑ Simplified APB/AHB BUS structure
- ❑ AuFPU and CAN bus are optional plug-ins



From Cache to Addressable Local RAM

- ❑ **Cache tries to keep frequently accessed information close to the processor for low latency accesses**
- ❑ **Cache is an expansive mechanism that may be not effective in specific embedded application:**
 - Data are not always reused, or have limited reuse
 - Data are loaded only when requested the first time
 - Cache line may be not a good match for data size
 - In Real-Time applications, Cache makes the system behavior not easily predictable
- ❑ **In micro-controller the complexity is moved from HW to SW**
- ❑ **Micro-controller is expected to execute few programs heavily optimized by hand**
- ❑ **The programmer is in charge to move information close to the processor when it is needed**

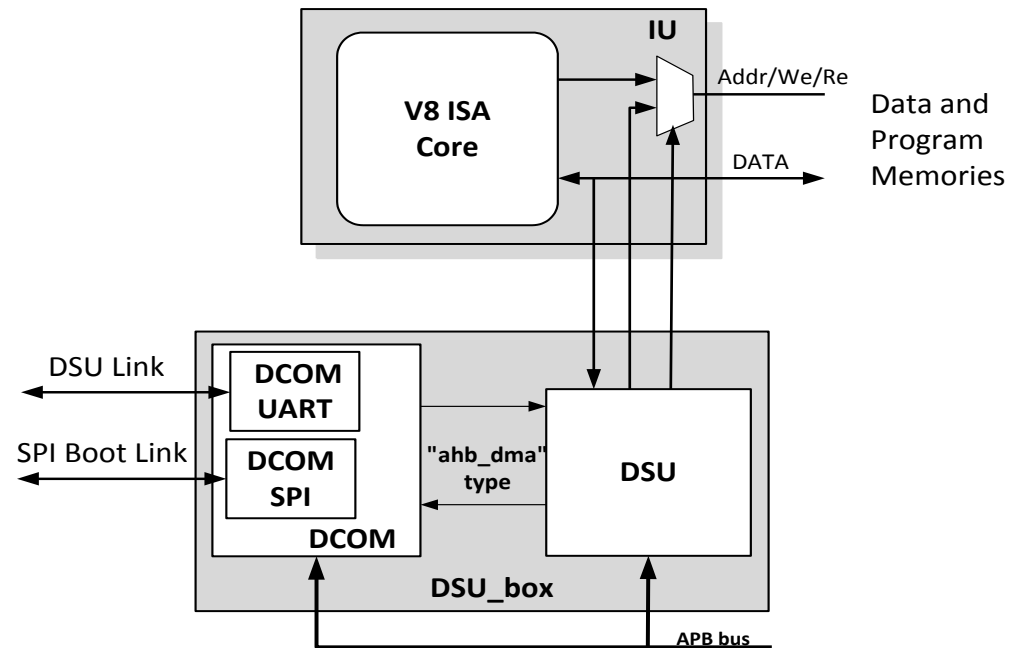
Peripherals

- ❑ **New functionalities are added to the LEON2 peripherals to address the largest range of devices without need of external logic (FPGA)**
 - Up to 4 x 16 Timers/PWM
 - Master/slave SPI 8 or 16 bit function
 - 4 x 24 bit bitwise programmable GPIO
 - Interrupt and Sleep Controller (Each peripheral and the processor pipeline itself have an 'Enable' that may be implemented as stretch control for the clock line. Disabled blocks go into low-power mode with no switching activity)

Debug Support Unit

- ❑ **V8uC partially reuses DSU modules of the LEON2-FT**
- ❑ **The DSU box module allows to know the internal state of V8uC microcontroller through its Debug Serial Link**

- A new SPI module is included to allow the fetch of DSL commands from an external Serial Memory as boot option
- As alternative boot option program image can be directly fetched into local RAM by the DMA engine



Software Development Tools: compiler and debugger

- ❑ V8uC is a Sparc V8 processor compliant with the GNU sparc-elf-gcc cross compilers
- ❑ The core is equipped with boot code, linker scripts and V8uc-GDB stub file for the easy 'C' programs compiling and debugging with GNU tools
- ❑ The DSU module has been maintained for compatibility with LEON2-FT Tools
- ❑ Except for memory controller and cache settings, V8uC can execute LEON2-FT codes without changes

DDD: /projects/letsme09/LMC_FPGA_DB/SW/tsource/dumb_main.c

File Edit View Program Commands Status Source Data Help

((): g1

1: res 0 2: flag 33 3: flag == 33 && i == 1 0

```
56         flag=33;
57     }
58
59
60     for (i=0;i<30;i++)
61     {
62         flag=flag+i;
63     }
64
65
66
```

0x400013c0 <main+28>: st %g1, [%fp + -4]
0x400013c4 <main+32>: ld [%fp + -4], %g1
0x400013c8 <main+36>: inc %g1
0x400013cc <main+40>: st %g1, [%fp + -4]
0x400013d0 <main+44>: ld [%fp + -4], %o0
0x400013d4 <main+48>: call 0x4000133c <fact>
0x400013d8 <main+52>: nop
0x400013dc <main+56>: mov %o0, %g1
0x400013e0 <main+60>: st %g1, [%fp + -8]
0x400013e4 <main+64>: clr [%fp + -12]

(gdb) target remote devttyUSB0
devttyUSB0: No such file or directory.
(gdb) target remote /dev/ttyUSB0
0x40003758 in breakinst () at sparc-stub.c:1021
/projects/letsme09/LMC_FPGA_DB/SW/tsource/sparc-stub.c:1021:27116: beg:0x40003758
(gdb) next
(gdb) next
main () at dumb_main.c:35

Display 1: res (enabled, scope main, address 0x4001ffe8)

Core Numbers: Dhrystone benchmark

- ❑ The Dhrystone benchmark for integer computation and strings manipulation has been executed on a V8uC, LEON2-FT and M8051
- ❑ V8uC: 8 Register windows; 64Kbyte program + 64Kbyte data memories on chip
- ❑ LEON2 (with cache enabled): Instruction Cache = 4 x 8Kbyte x 8byte; Data Cache = 2 x 8Kbyte x 4byte

FPGA	V8uC	LEON2-FT	MC8051
Clock Frequency	32 MHz	32 MHz	16 MHz
Dhrystone/s	49230	44076	595
MIPS/MHz	0.876	0.780	0.021

- ❑ V8uC is 10% faster than LEON (Local Memory vs Caches advantage) and 40 times faster than M8051 (8 bit are too few to manage Dhrystone bench !?)

Conclusions

- ❑ **V8uC core passed the SPARC V8 Compliance Test Suite and additional test procedures for its peripherals**
- ❑ **A specific stress test produced all the possible sequences of 5 consecutive memory instructions also in condition of fault injection and simultaneous memory washing by the scrubber unit**
- ❑ **Further optimizations are possible:**
 - on HW: Trap-Table, Code compression and IRQ management
 - and SW: Microcontroller optimized libraries, Instruction Set Simulator



V8uC Demo Box

Final Conclusions

Final Conclusions (1)

❑ **To adopt IP Core design rules like:**

- Name convention
- File Header
- Ports ordering and mapping
- Coding Practices
- Portability
- Use of clock and reset
- Synchronicity
- Combinational and Sequential Block
- Coding for Synthesis
- Partitioning
- Coding for DFT

Final Conclusions (2)

❑ **IP Core Verification**

- To ensure the IP Cores is 100% correct in its functionality and timing
- Testbenches and test suites must be reusable by others teams and compatible with verification tools

❑ **Verification Strategy**

- Individual sub-block
- Macro verification
- Prototyping

❑ **Verification Tools**

- Simulation
- Testbench automation tools
- Code coverage tools

Final Conclusions (3)

- ❑ **To define a common validation procedure**
 - RTL and back-annotated simulations
 - Software simulations
 - Hardware test on re-programmable FPGAs
 - Hardware test on EM FPGA (like Microsemi AX devices)
- ❑ **High level of configuration/programmability can cause issues on IP cores validation**
 - Indeed back-annotated simulations can cover only a limited number of possible IP core configurations
 - STA to verify timing closure in case of fully synchronous design
 - Back-annotated simulations to verify external interface, clock domains (if any) and reset distribution
- ❑ **Could it be useful an ECSS standardization to cover the IP core development?**
- ❑ **To define a strategy for maintenance and long-term support**



***Thank you for your
attention!***

Franco Bigongiari

Phone: +39 050 9912116

E-mail: franco.bigongiari@sitael.com

URL - <http://www.sitael.com>

SITAE L S.p.A.

S. P. 231, KM. 79.900 - 70026 Modugno (BA)

Via Livornese 1019 - 56122 Pisa (PI)

ITALY

<http://www.sitael.com>