COBHAM

The most important thing we build is trust

ADVANCED ELECTRONIC SOLUTIONS | AVIATION SERVICES | COMMUNICATIONS AND CONNECTIVITY | MISSION SYSTEMS

Design and Implementation of an Inter-Processor Link (RapidIO) for Future OBCs

Contract number: 4000112902/14/NL/LF

Final Presentation

TEC-ED & TEC-SW Final Presentation Days – May 2017

Date: 09.05.2017

Speaker: Stefano Di Mascio (Cobham Gaisler AB)

Technical Officer: Kostas Marinis (TEC-EDD)

Commercial in confidence

Table of Contents

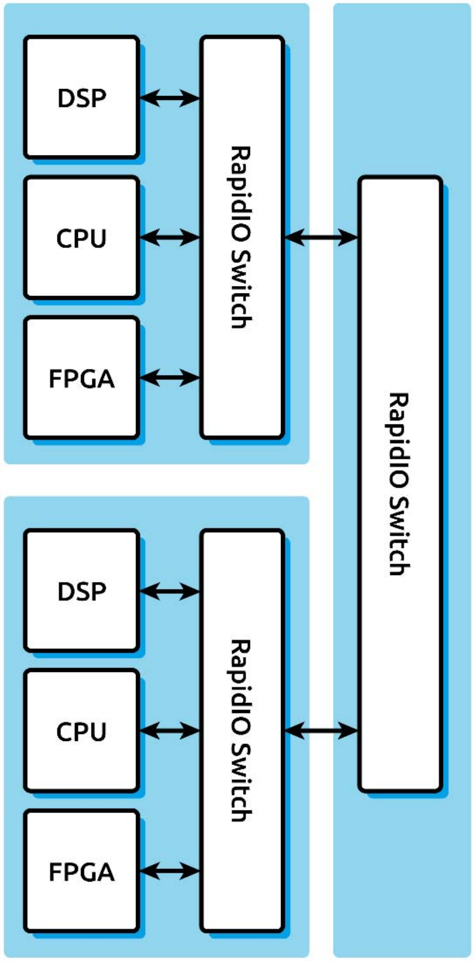
Final Presentation: Inter-processor Link for Future OBCs

- Introduction to Serial RapidIO
- SRIO Logical Layer IP core (GRSRIO)
- Verification
  - Stand-alone testbench
  - Integration testbench
- Estimated synthesis and place & route results
- Validation
  - FPGA prototypes
  - Software and results
  - RUAG Space AB integration test
- IP-XACT Model
- Experience from activity
- Conclusions

Introduction to Serial RapidIO

Rationale for RapidIO as an Inter-processor Link

- Future high-performance on-board computers will require **fast and reliable** communication links for:
  - **processor to processor** communication
  - integration of other complex chips (e.g. **co-processors**)
- Available solutions on the market are:
  - Low-speed serial interfaces
    - Not fast enough
  - Parallel buses
    - Large pin counts
    - Do not scale well
  - High-speed serial interfaces
    - PCI express
      - Not well suited for multi-processor systems
    - **Serial RapidIO** (SRIO)
      - Designed for **multi-processor** systems



Introduction to Serial RapidIO

RapidIO Overview

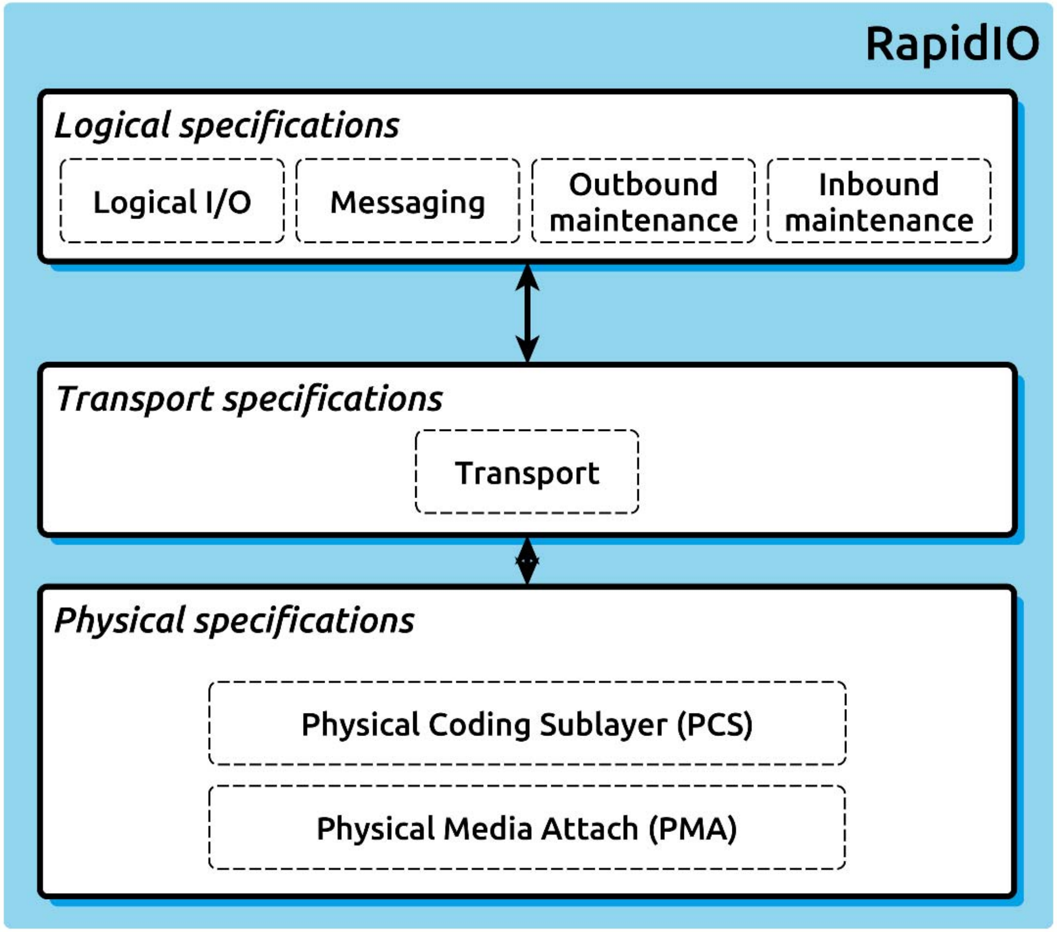
RapidIO is a **packet-switched** interconnect technology for chip-to-chip and board-to-board communication through a backplane

- Open standard (**www.rapidio.org**)
- Initially designed for signal processing, networking, and communications applications:
  - More than 10 years of market deployment
  - More than 100 million 10-20 Gbps ports shipped
    - ~100% 4G/LTE interconnect market share
    - ~60% Global 3G interconnect market share
- Interest in recent years from **military** and **aerospace** applications
  - Chosen by NASA/U.S. Air Force as preferred high-speed protocol for the next generation High Performance Spaceflight Computing (**HPSC**) processor
  - **N**ext **G**eneration **S**pace **I**nterconnect **S**tandard (U.S. govt./industries collaboration)
    - Included in **SpaceVPX** specifications together with SpaceWire and I2C
    - Specific fault tolerant extension for space added (from Rev. 3.1 onwards)
  - **Rad-hard** products by BAE Systems and Honeywell available

Introduction to Serial RapidIO

Three layers specifications (**Rev. 2.1**)

- Logical layer
  - Direct I/O operations: NREAD, NWRITE(_R), SWRITE, ATOMIC operations
  - Data messages (payload up to 256 B)
  - Doorbell messages (2 B)
  - Maintenance messages
- Transport layer
  - ID based
- Physical layer
  - Parallel RapidIO
    - 8/16b over LVDS (deprecated in newer revisions)
  - **Serial RapidIO**
    - From 1.25 up to 6.25 Gbps per lane (newer revisions up to 25 Gbps per lane)
    - Up to 16 lanes



Introduction to Serial RapidIO

Serial RapidIO Overview – Main advantages

- Meant for applications where **lost packets** and **undetected errors** are not tolerable (node to node reliable flow control at **physical layer**):
  - Strict **acknowledgement** scheme
    - A packet gets lost -> a timeout will expire on the transmit side and the packet will be transmitted again
  - **Retry** mechanisms (e.g. if a receiving node has no space to store the packet)
  - **Error coverage** (CRC16), if an error is spotted:
    - Synchronization between sender and receiver is verified
    - The packet is transmitted again
  - **Deadlock** situations prevented by a strict control and reordering based on **priority** levels

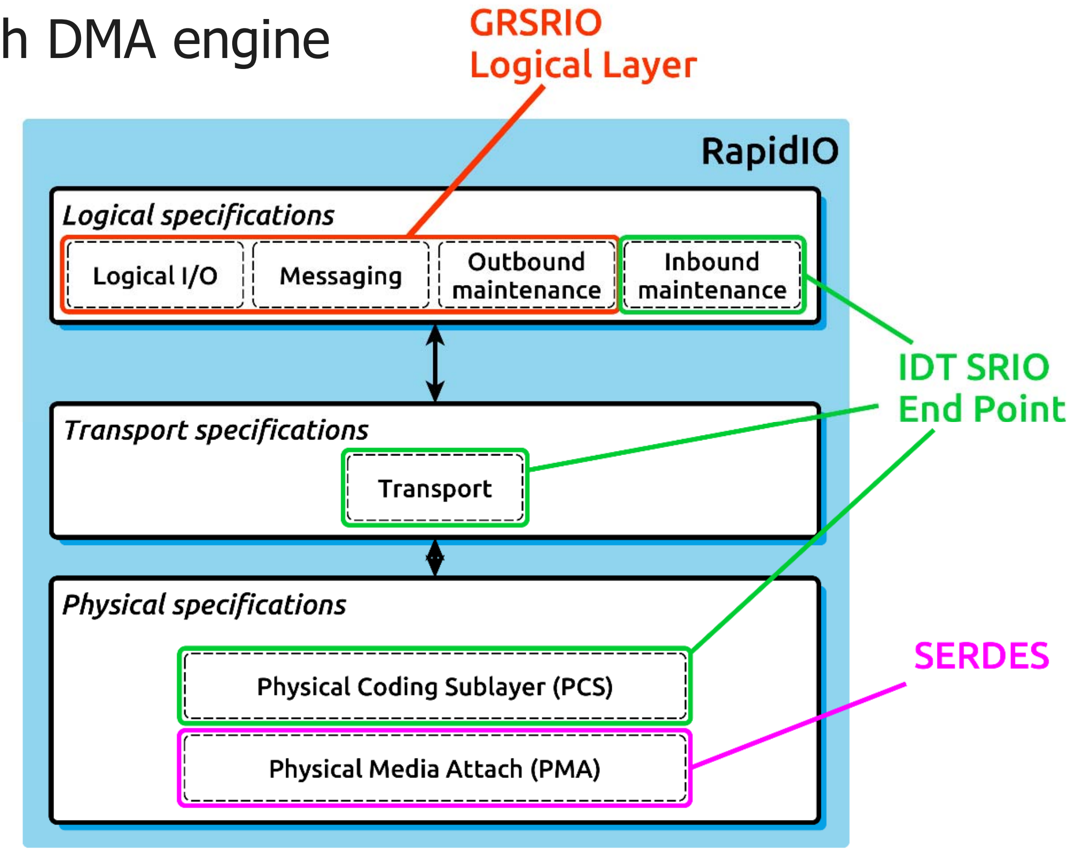
- Highly **scalable** network
  - True peer-to-peer communication
  - Point-to-point topology
  - Routed by crossbar switches

- Low **latency**
  - Full duplex nature
  - Distributed arbitration
  - High data signalling rates

SRIO Logical Layer IP Core (GRSRIO)

Our Solution

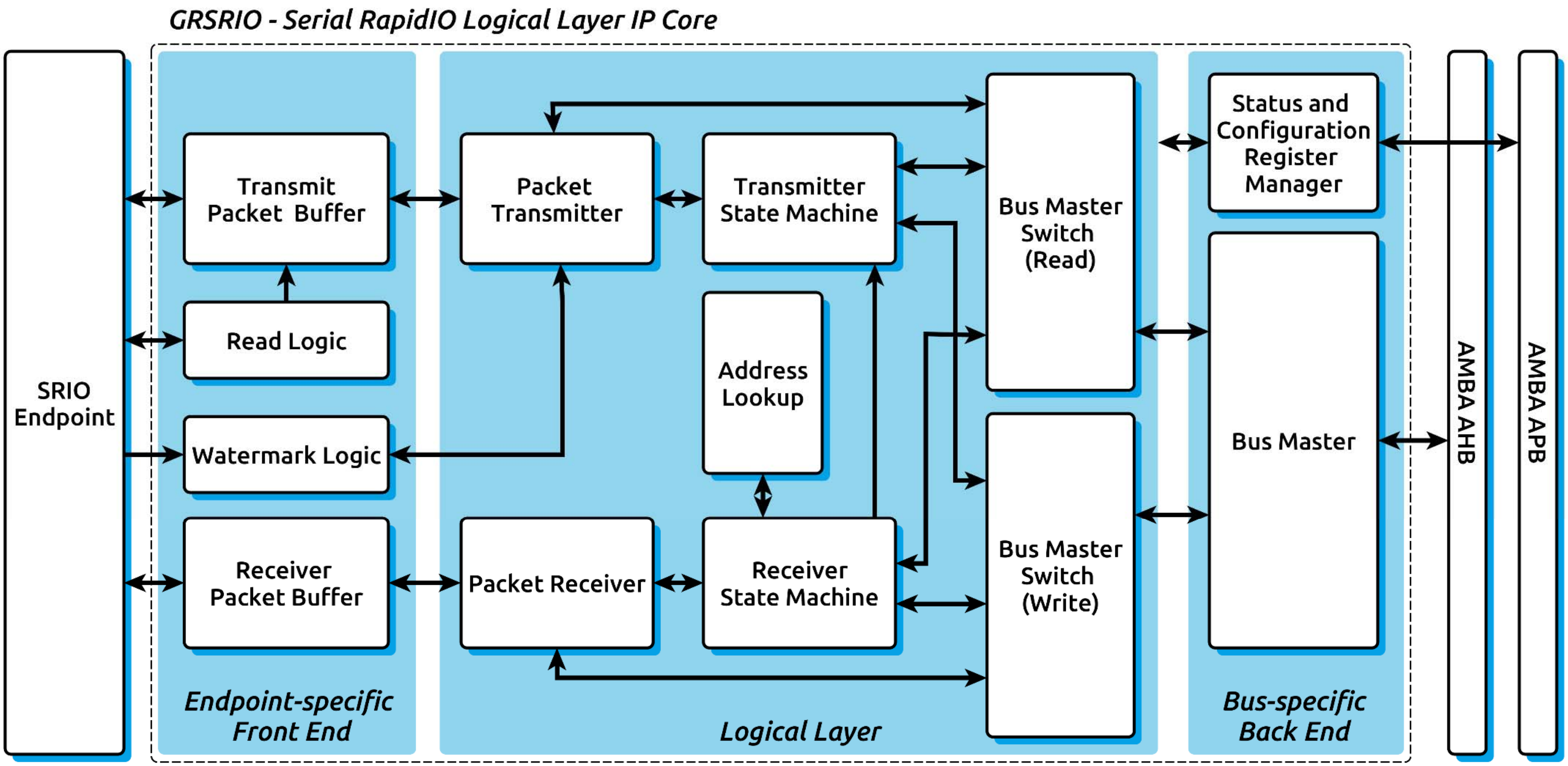
- We provide a **Logical Layer** with DMA engine
  - Logical layer implemented in HW (VHDL)
  - Supports **data messages**, **doorbell messages**, **outbound maintenance messages** and **I/O operations** as defined in I/O and Message Passing Logical Specification (**Rev. 2.1**)



- Based on the **SRIOIP-GEN2** by Integrated Device Technology (IDT)
  - One of the market leaders
  - Logical specifications not fully implemented (usually implemented in SW)
  - Up to 4 lanes
- Solution targeted at **ASICs**

SRIO Logical Layer IP Core (GRSRIO)

Architecture



Highly **modular** design:

easy to adapt to different SRIO endpoints and busses

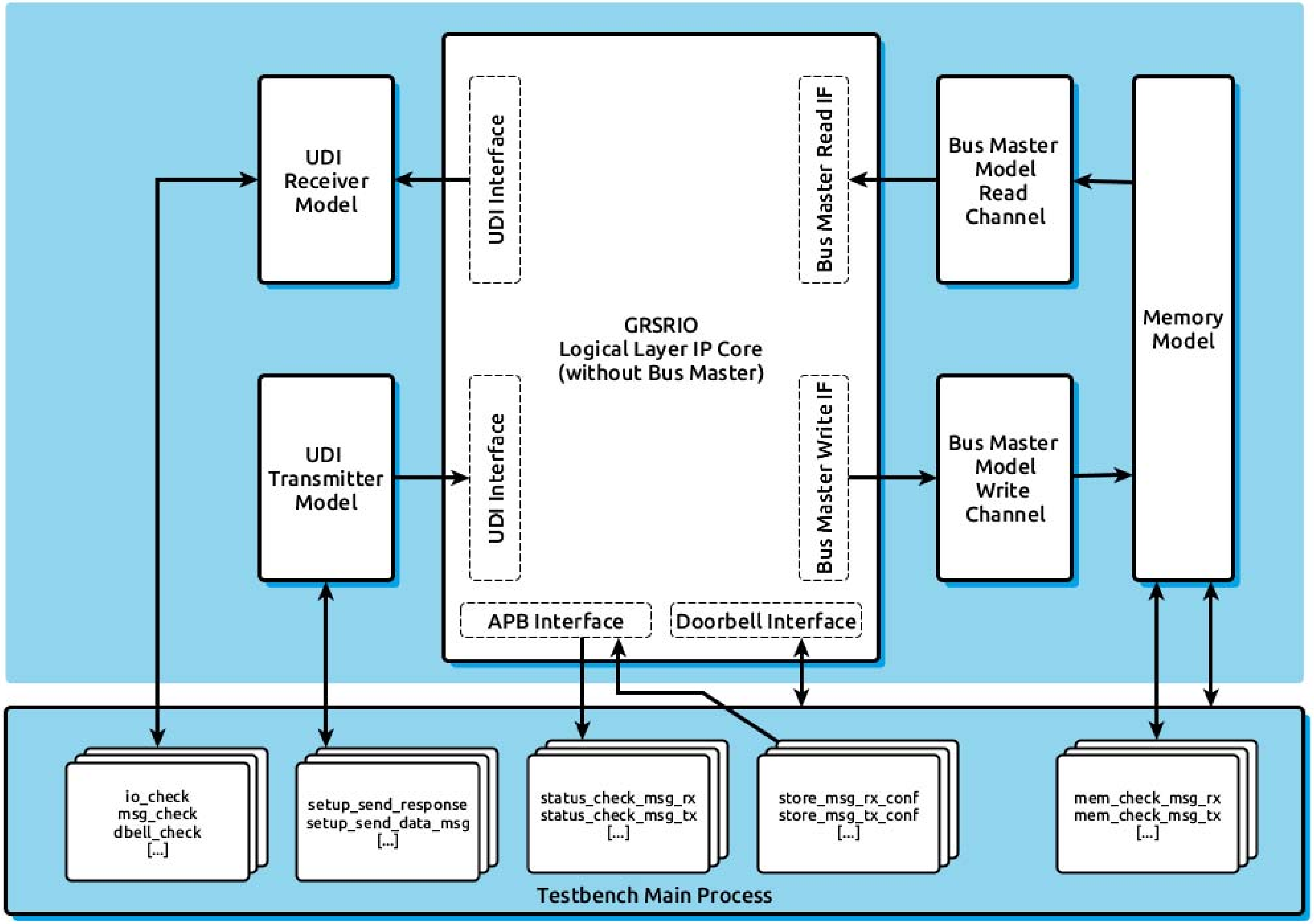
SRIO Logical Layer IP Core (GRSRIO)

Feature Set

- Configurable number of separate transmission and reception queues for messages
  - Throughput/area trade-off
- Storage size and base address of the transmission and reception queues can be dynamically changed at run time
  - Efficient memory utilization
- An **external doorbell interface** allows the generation and reception of doorbell messages by external hardware components
  - e.g. to trigger interrupts in the receiving End Node with minimum **latency**
- **Multi-segment messages** are automatically assembled and disassembled by the IP core
  - CPU offloading
- Inbound data and doorbell messages can be **filtered** or routed to reception queues based on their **destination ID** and/or **mailbox number**
- Inbound I/O operations can be restricted by means of four **memory partitions** with configurable size and **write protection** (**reliability**)
- AMBA 2.0 **AHB master** interface with configurable bus width and burst length and **APB slave** interface

Verification

Stand-alone testbench: Overview



Self-checking testbench achieving full code coverage

Verification

Stand-alone testbench: test cases and results

Doorbell messages to the GRSRIO core

Send doorbell messages to the GRSRIO core. All conditions tested (successful, timeout, automatic retry, error response, etc.). External doorbell interface tested.

Data messages to the GRSRIO core

Send multi-packet and single-packet data messages to the GRSRIO core with all possible size and conditions (successful, timeout, retry, error, multiple queues, etc..)

I/O operations to the GRSRIO core

Send all types of I/O operations to the GRSRIO core with all possible sizes and conditions (timeout, bus error, etc.) to the GRSRIO core

Miscellaneous

Test of the Receiver Packet Buffer, Transmission Packet Buffer, Watermark logic to interface the End Point and debug registers

Doorbell messages from the GRSRIO core

Send doorbell messages from the GRSRIO core. All conditions tested (successful, timeout, automatic retry, error response, etc.). External doorbell interface tested.

Data messages from the GRSRIO core

Send multi-packet and single-packet data messages from the GRSRIO core with all possible sizes and conditions (successful, timeout, retry, error, multiple queues, etc..)

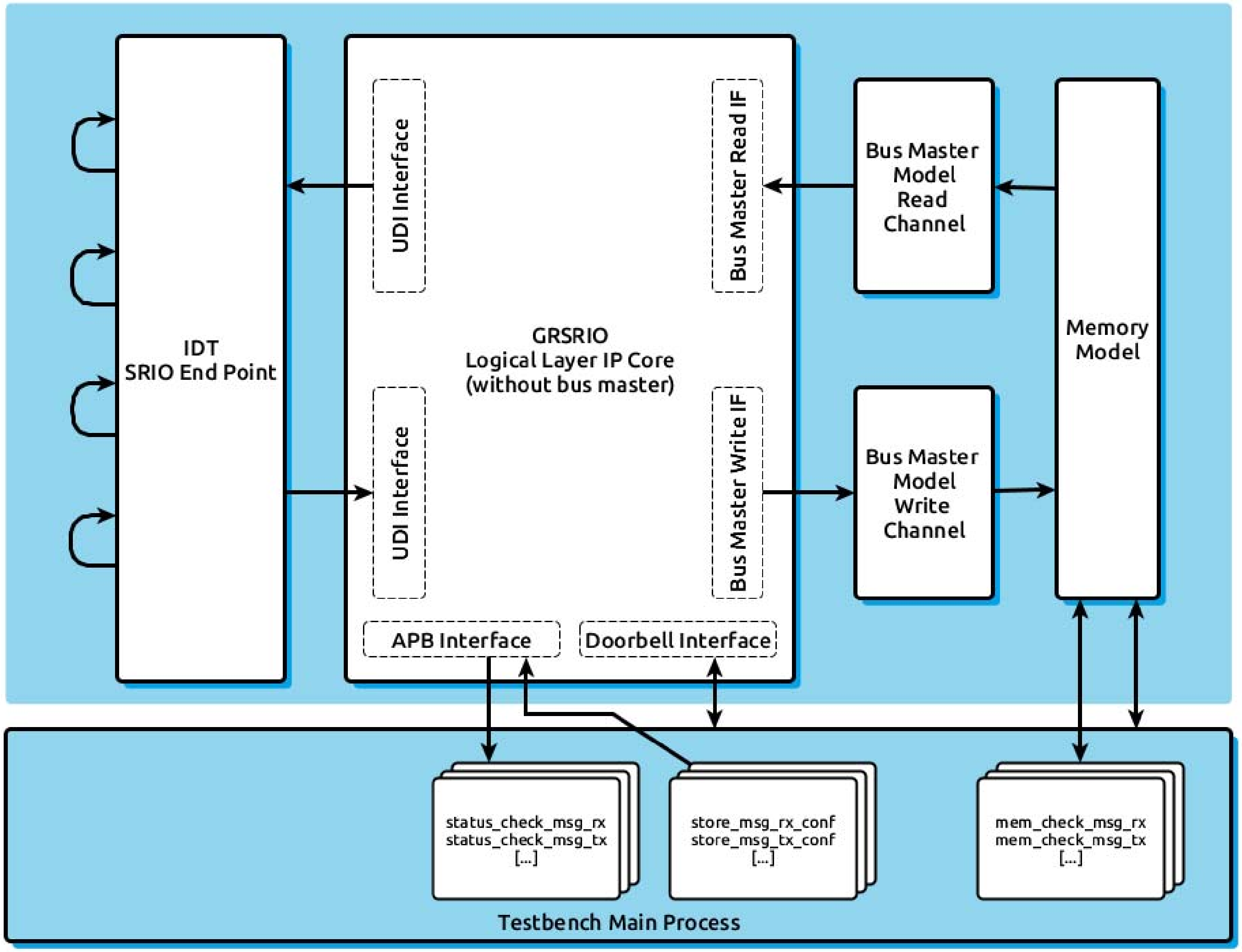
I/O operations from the GRSRIO core

Send all types of I/O operations from the GRSRIO core with all possible sizes and conditions (timeout, bus error, etc.) from the GRSRIO core

- All tests **passed**
- 100% statement and branch code coverage achieved for all source code

Verification

Integration testbench



Proof of correct integration with IDT SRIO End Point

Synthesis and place & route results

Rad-hard 65 nm ASIC and Xilinx Virtex-7

Rad-hard 65 nm ASIC technology for space synthesis estimates[1]

- Frequency:
  - 6.25 Gbps per lane slightly missed
    - 307.7 MHz achieved vs. 312.5 MHz required for the clock of the IDT End Point (should be attainable with further efforts and optimizations in the synthesis)
    - uncertainties in **pre-layout estimates** (wire-load models)
  - Big margin on 5 Gbps per lane
- Area:

Component	Gate Equivalents
GRSRIO Logical Layer	34,511
IDT End Point + GRSRIO	189,019

Xilinx Virtex-7 PAR results[2]

- Frequency:
  - >1.25 Gbps per lane
- Area:

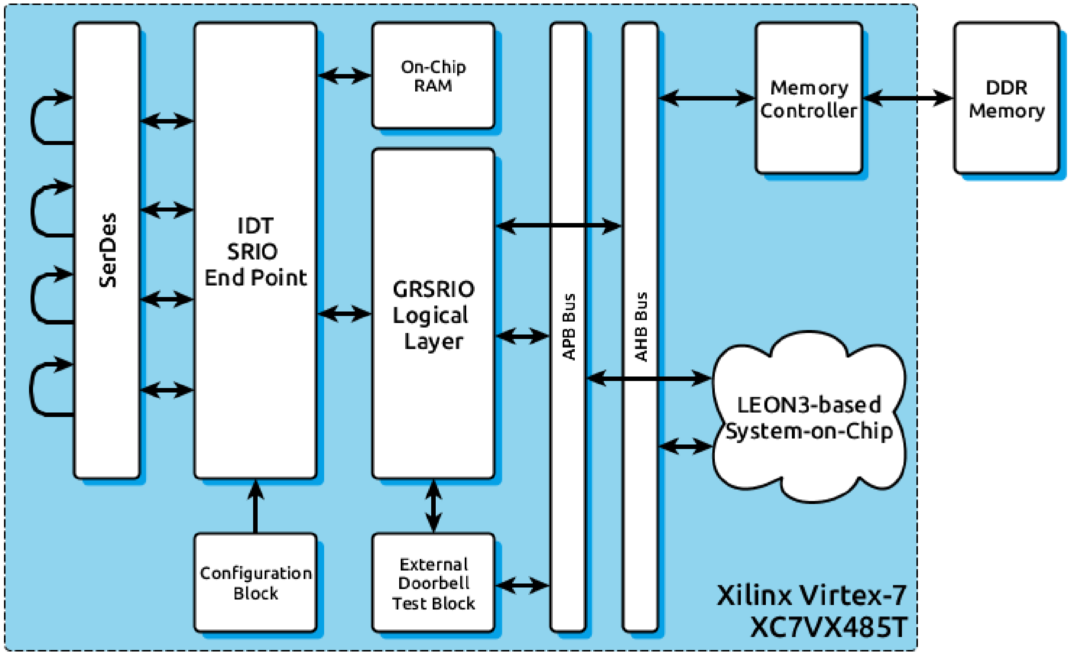
Component	LUTs	Registers	BRAM
GRSRIO Logical Layer	7,842 (2.6%)	5269 (0.9%)	9 (0.9%)
IDT End Point + GRSRIO	42084 (13.9%)	21658 (3.6%)	49 (4.8%)

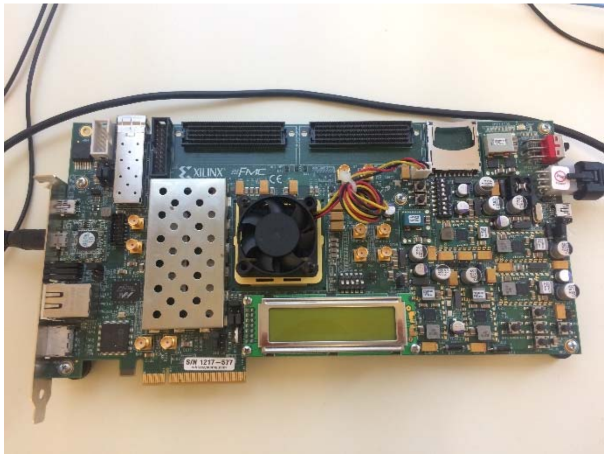
1) Estimated by Design Compiler. All RAMs black-boxed. Gate equivalent based on NAND2 (5.2 µm^2).

2) PAR by Vivado toolchain. FPGA: Virtex-7 XC7VX485T-2.

Validation

FPGA Prototypes





- Two FPGA boards have been used to implement the validation testbench:
  - Xilinx **Virtex-7** FPGA VC707 Evaluation Kit
    - Successful implementation of the whole validation setup (1.25 Gbps x 4 lanes: 6 Gbps port)
  - Microsemi **RTG4** Development Board
    - SerDes could not be integrated due to the minimum frequency required (100 MHz)

Validation

Software and results

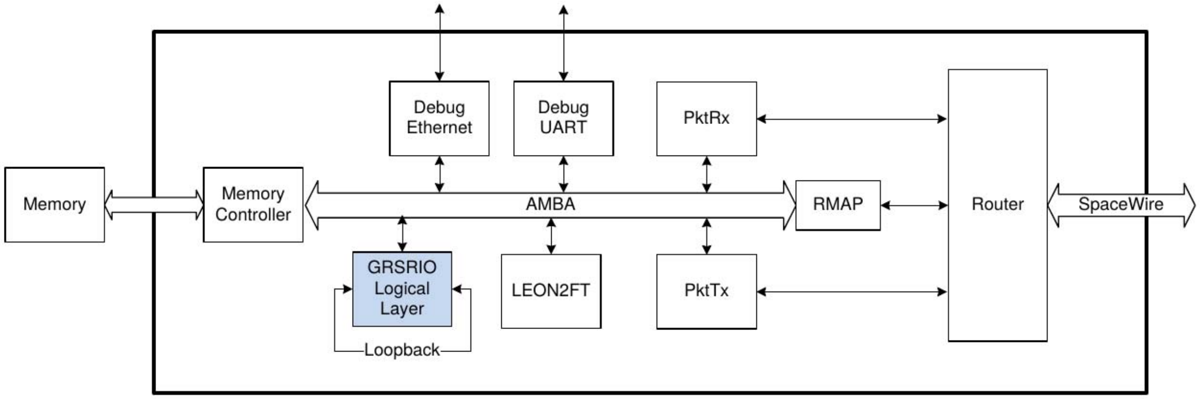
- Bare-C software for LEON3 microprocessor
- Extensive API including compare functions to build self-checking tests
- Most parameters randomized
- All tests successfully executed on both Virtex-7 and RTG4

Test Cases		Result
1	Send doorbell messages to/from 4 buffers	PASSED
2	Send single-segment messages to/from 4 queues	PASSED
3	Send single- and multi-segment messages to/from 1 queue	PASSED
4	Write data to memory using NWRITE_R transactions	PASSED
5	Read data from memory using NREAD transactions	PASSED
6	Modify data in memory using atomic transactions	PASSED
7	Generate and receive doorbells through external interface	PASSED
8	Write and read data using maintenance packets	PASSED

Validation

RUAG Space AB integration test results

RUAG Space AB (Sweden) carried out an **independent** validation campaign by integrating the GRSRIO into one of their LEON2-FT-based SoCs (VHDL simulation)



Test scenarios	
1	Transmission of two doorbell messages from one doorbell buffer to two different doorbell reception buffers
2	Test the transmission of write and read accesses using the I/O operation functionality
3	Transmission of data through the following path: 1. SpaceWire (PktRx) 2. Memory 3. GRSRIO Message Transmission Buffer 4. GRSRIO Message Reception Buffer 5. Memory 6. SpaceWire (PktTx)

All scenarios successfully tested!

IP-XACT Model

Overview

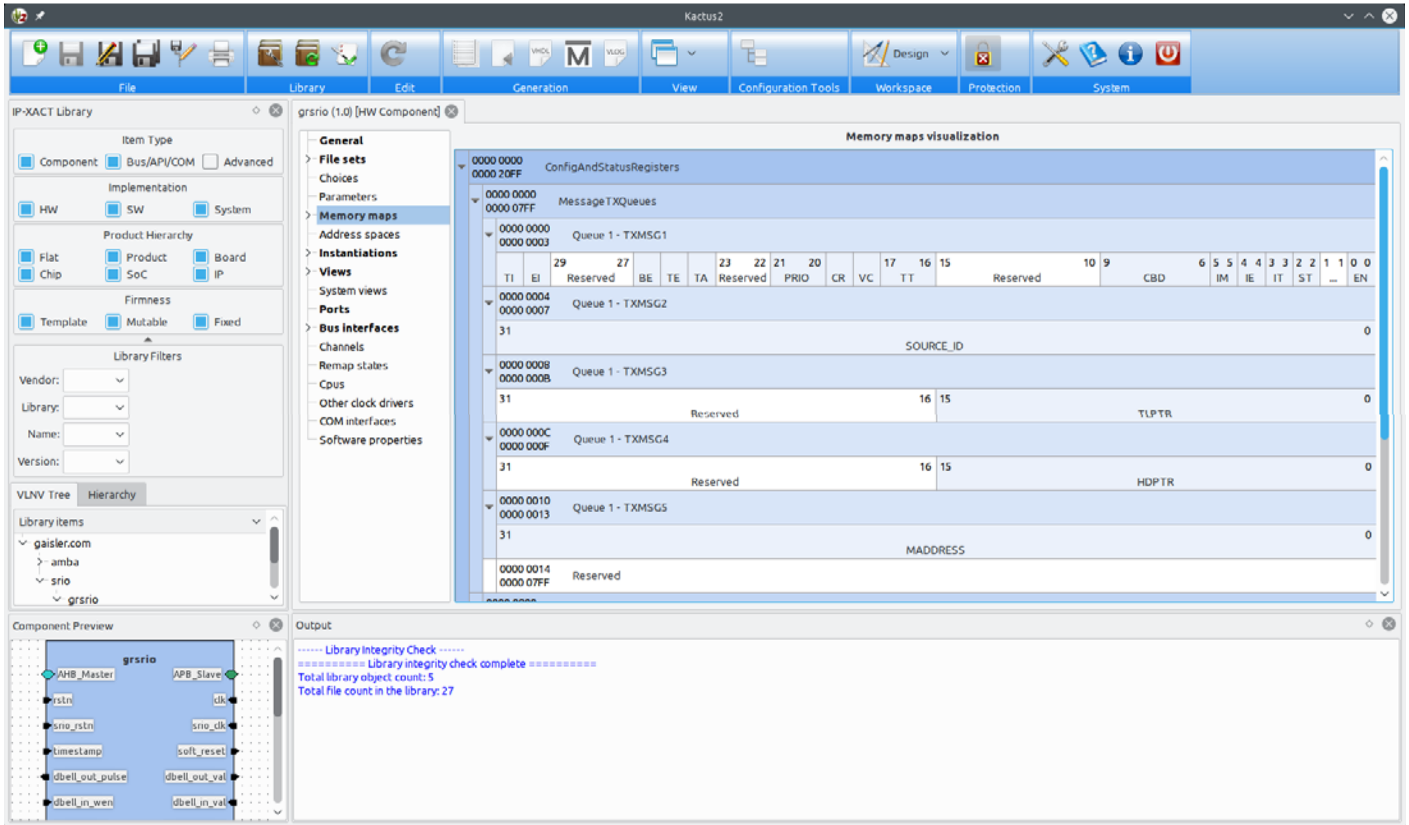
- **Vendor-independent** description for IPs based on XML
  - Memory maps and registers
  - Ports and bus interfaces
  - Configuration parameters (generics)
  - File sets (dependencies)
- Defined by an IEEE Standard (IEEE 1685-2014)
- Increases automation for IPs selection, configuration and integration
  - Manage increasing design **complexity**
- Optimizes **multi-vendor** SoC design flow from architectural design to chip layout
  - Shorter time to market/lower development cost
  - Vendor-independent tools and scripts (e.g. document generation, block-diagrams IP integration, etc.)
- Eases the handling of internal and external IP libraries for IP providers
- The standard allows optional vendor-specific extensions that can harm direct compatibility

IP-XACT Model

Kactus2: Example of an EDA tool based on IP-XACT models

Open source GUI-based EDA tool by Tampere University of Technology

- Import or create IP-XACT models
- Create HW designs by instantiating, configuring and interconnecting component instances in a graphical way
- Generate HDL files with wiring and parameterization
- Manage memory maps and address spaces
- Generate Makefiles



```
------ Library Integrity Check ------
========== Library integrity check complete ==========
Total library object count: 5
Total file count in the library: 27
```

Kactus2 available at http://funbase.cs.tut.fi/

Experience from activity

Lessons learned and proposed improvements

Valuable experience in developing high-speed interfaces for space

- **Space-grade** FPGAs require a tailored SRIO End Point to achieve reasonable throughputs
- High-speed serial links will benefit from higher performance **on-chip busses**
  - Moving from AHB-based SoCs to **crossbar-based** SoCs (e.g. AXI4)
    - Several bus masters can use the bus simultaneously
    - Several outstanding read transactions on the bus from the same master
    - Can typically achieve longer bursts
- **Memories** are usually the **bottleneck**
  - Extend DMA descriptors to define more than one I/O operation with contiguous payload, to avoid **dead times** due to the opening and closing of descriptors when transmitting contiguous data
    - More than one **outstanding** RapidIO operation per queue to enhance performance of operations with response

Conclusions

We have developed a flexible **logical layer** for Serial RapidIO (**GRSRIO**)

- **Modular**
  - Easily adaptable to different End Points (e.g. Xilinx or Altera IP cores) and different busses (e.g. AXI4)
- **Extensively verified**
  - 100% statement and branch coverage
- Validated by means of two **FPGA prototype** platforms
  - Full **Memory-SerDes-Memory** loopback successful with Virtex-7 on an AHB-based SoC (1.25 Gbps per lane x 4 lanes)
- Designed for and delivered with the **SRIOIP-GEN2 End Point by IDT**
  - Reference implementation used in many **commercial** applications
  - Targeted at **ASICs**
    - includes a big set of optional functionalities and very extensive debug features
- **GRSRIO and SRIOIP-GEN2 freely licensed for ESA projects**
  - All three layers of the RapidIO protocol **implemented in hardware**
    - Minimum CPU loading

- Thank you -

For questions please contact:
support@gaisler.com