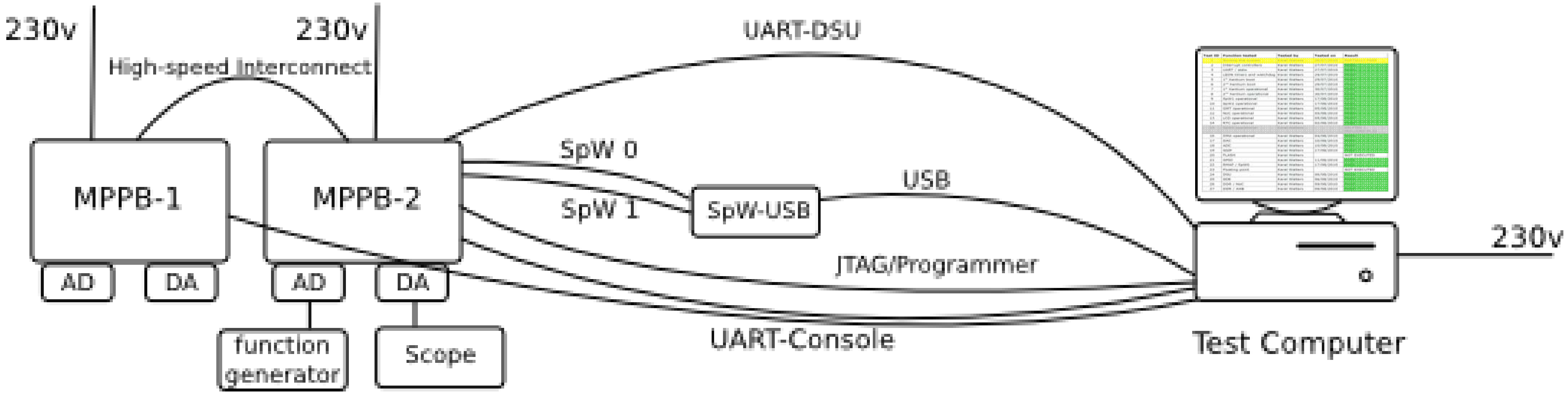
Outline

- MPPB platform hardware
- MPPB / Xentium® software development
- MPPB validation & benchmarking
- MPPB FDIR
- ASIC migration & radiation hardening
- MPPB lessons learned / future directions
- MPPB demonstration
- Q&A

Validation Test Procedures



- All Validation tests passed
- Endurance tests run with IO interfaces (>72 hours)

Floating-point calculations on Xentium DSP
Programmers' view

- XentiumTools
  - Floating Point (FP) software emulation
  - FP support completely integrated in XentiumTools (by default)
- ANSI C (C99) standard programming language
  - Native floating point types and operations
  - Standard C math libraries (eg. libm)
- IEEE-754 compatibility
  - One rounding mode: round to nearest, ties to even
  - Denormalized numbers handled
  - Simplified exception handling
    - No interrupts and no status flag

Floating-point calculations on Xentium DSP
Software emulation implementation

- Xentium compiler translates C programs using floating-point into programs using only integers
  - Compiler replaces FP operations and conversions by calls to FP emulation functions using only integers
- FP emulation functions
  - based on C library from the LLVM Compiler Infrastructure Project
- Standard C-libraries
- Performance depends on efficient compiler and libraries
  - Cycle count and code size

MPPB I/O

- I/O performance

ADC-NoC	40 MS/s	*Limited by ADC performance*
DAC-NoC	40 MS/s	*Limited by DAC performance*
SpW-NoC	70 Mbit/s	*SpW runs at 100Mbps (gross)*
SpW-RMAP	70 Mbit/s	*SpW runs at 100Mbps (gross)*
Gigabit I/F	1.1 Gbit/s	*Requirement; V-5 supports 6.5 Gbit/s*

- Running I/O concurrently at **maximum** speed can be achieved by using different memory resources:
  - Xentium data memories / SRAM memory tile connected to NoC
  - SDRAM external memory connected to NoC / AHB

MPPB Benchmarking

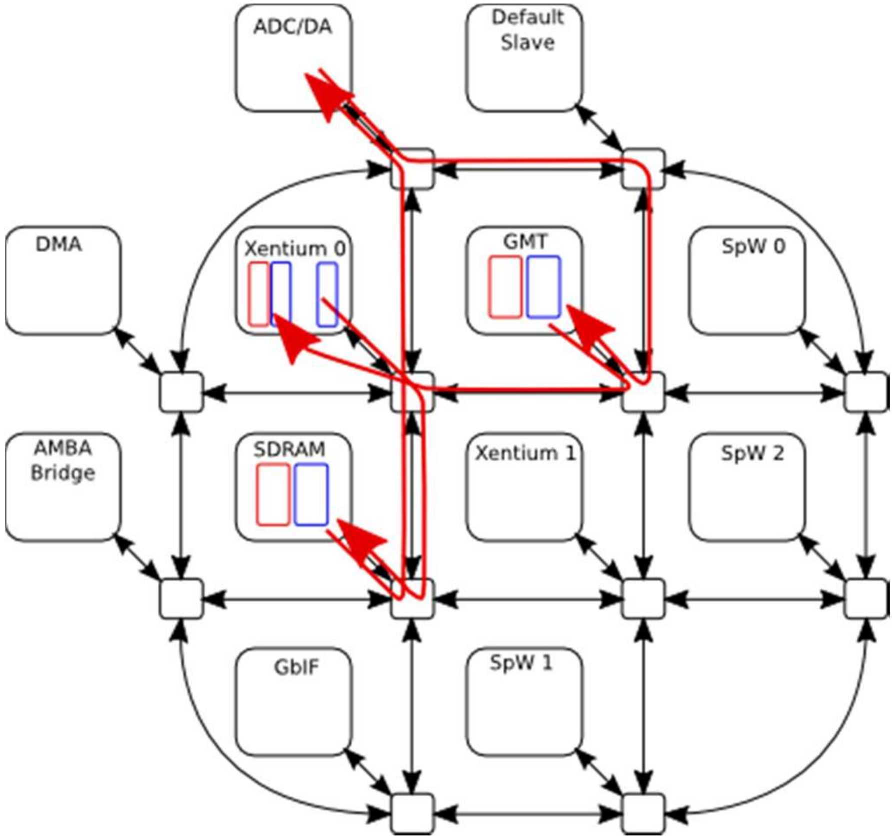
- **Analogue Data Acquisition Processing and Output**

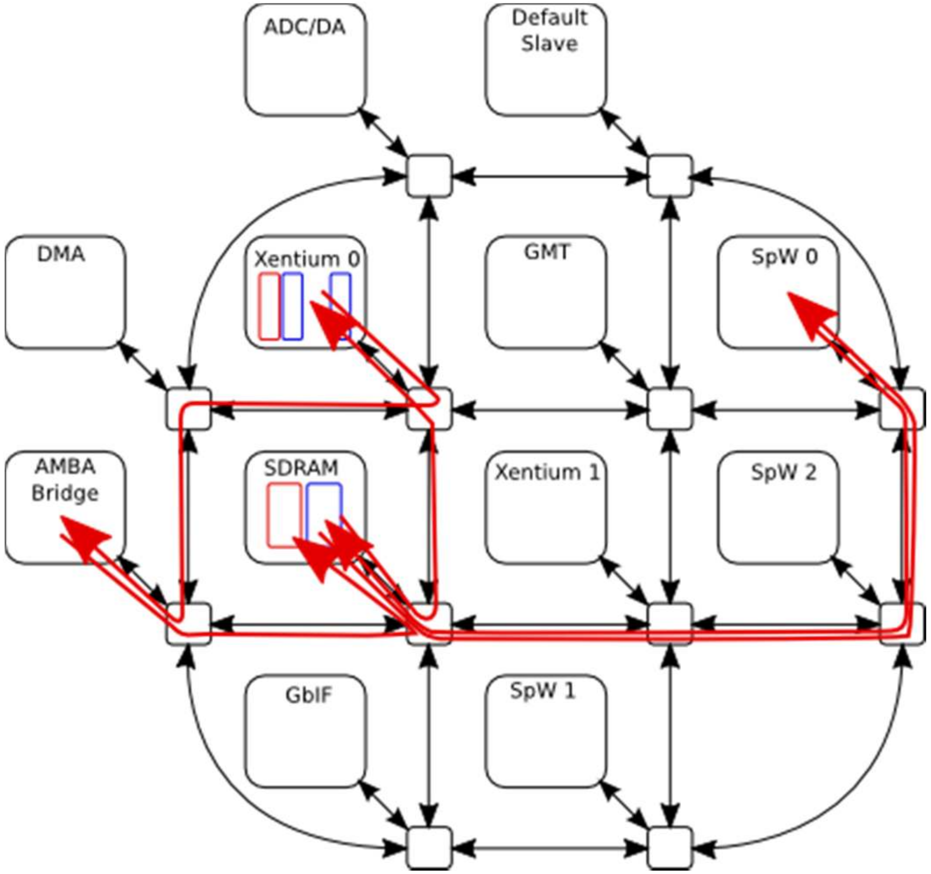
DAQ processing	*Max Data rate*	*FIR exec time (1024 samples)*	*FFT exec time*
Case 1: unprocessed	*40 MS/s*	*$2*10^{-5}$ s*	*NA*
Case 2: FIR LP 16	*6 MS/s*	*$1.7*10^{-4}$ s*	*NA*
Case 3: FIR LP 64	*2.5 MS/s*	*$4.1*10^{-4}$ s*	*NA*
Case 4: FIR LP 256	*730 kS/s*	*$1.3*10^{-3}$ s*	*NA*
Case 5: FFT1024	*6.46 MS/s*	*NA*	*$9.4*10^{-5}$ s*
Case 6: FFT1960	*4.73 kS/s*	*NA*	*0.2 s*
Case 7: FFT4096	*865 kS/s*	*NA*	*0.005 s*

- **Parallelism**
  - **Single Xentium used in the benchmark**
  - **code fairly mature, except for 1960 and 4096 FFT mapping**

MPPB Benchmarking

- ***Analogue Data Acquisition Processing and Output***



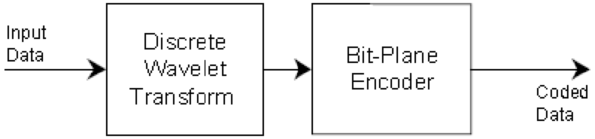


MPPB Benchmarking

- Image Data Compression
  - 2D DWT
  - Encoder
- Control code of DWT on Xentium developed in C
- DWT kernels on Xentium implemented in assembly
- Benchmark concerns are
  - large amount of bit operations and speed difference
  - Code only functional, not optimal, transform is fairly mature
  - Large speed gain could be made in the encoder ~4x

MPPB Benchmarking

- **Image Data Compression**

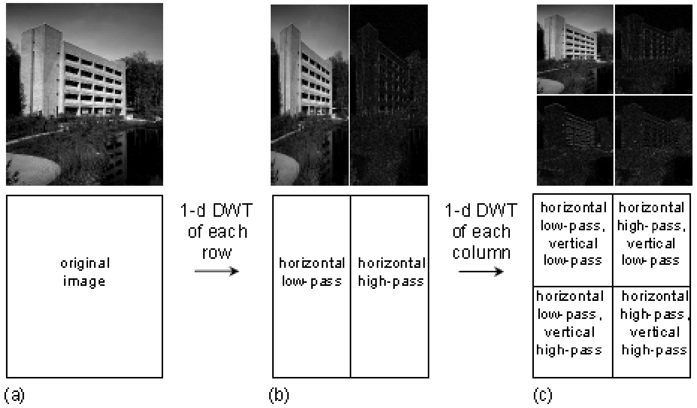


Input Data

Discrete Wavelet Transform

Bit-Plane Encoder

Coded Data



original image

1-d DWT of each row

horizontal low-pass

horizontal high-pass

1-d DWT of each column

horizontal low-pass, vertical low-pass

horizontal high-pass, vertical low-pass

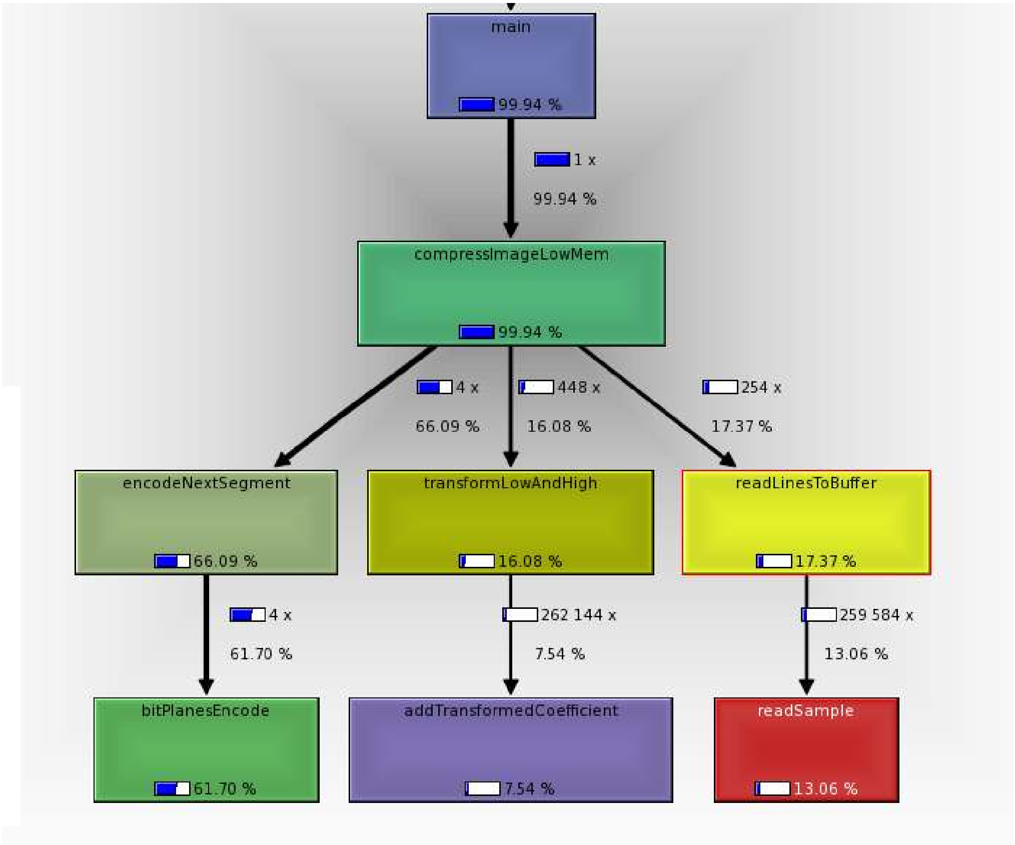
horizontal low-pass, vertical high-pass

horizontal high-pass, vertical high-pass

(a)

(b)

(c)



main 99.94 %

1 x 99.94 %

compressImageLowMem 99.94 %

4 x 66.09 %

448 x 16.08 %

254 x 17.37 %

encodeNextSegment 66.09 %

transformLowAndHigh 16.08 %

readLinesToBuffer 17.37 %

4 x 61.70 %

262 144 x 7.54 %

259 584 x 13.06 %

bitPlanesEncode 61.70 %

addTransformedCoefficient 7.54 %

readSample 13.06 %

RECORE

MPPB Benchmarking (Rice encoding)

Given a constant M, any symbol S can be represented as a quotient (Q) and remainder (R), where:

$$S = Q \times M + R.$$

If S is small (relative to M) then Q will also be small

Rice encoding represents Q as a unary value and R as a binary value.

A value N may be represented by N 1s followed by a 0.
Example: $3 = 1110$ and $5 = 111110$.

Given a bit length, K. Compute the modulus, M using by the equation $M = 2^K$.
Then do following for each symbol (S):

Write out $S \,\&\, (M - 1)$ in binary.
Write out $S >> K$ in unary.

Encode the 8-bit value 18 ($0b00010010$) when $K = 4$ ($M = 16$)

$S \,\&\, (M - 1) = 18 \,\&\, (16 - 1) = 00010010 \,\&\, 1111 = 0010$
$S >> K$
$18 >> 4 = 0b00010010 >> 4 = 0b0001$ (10 in unary)

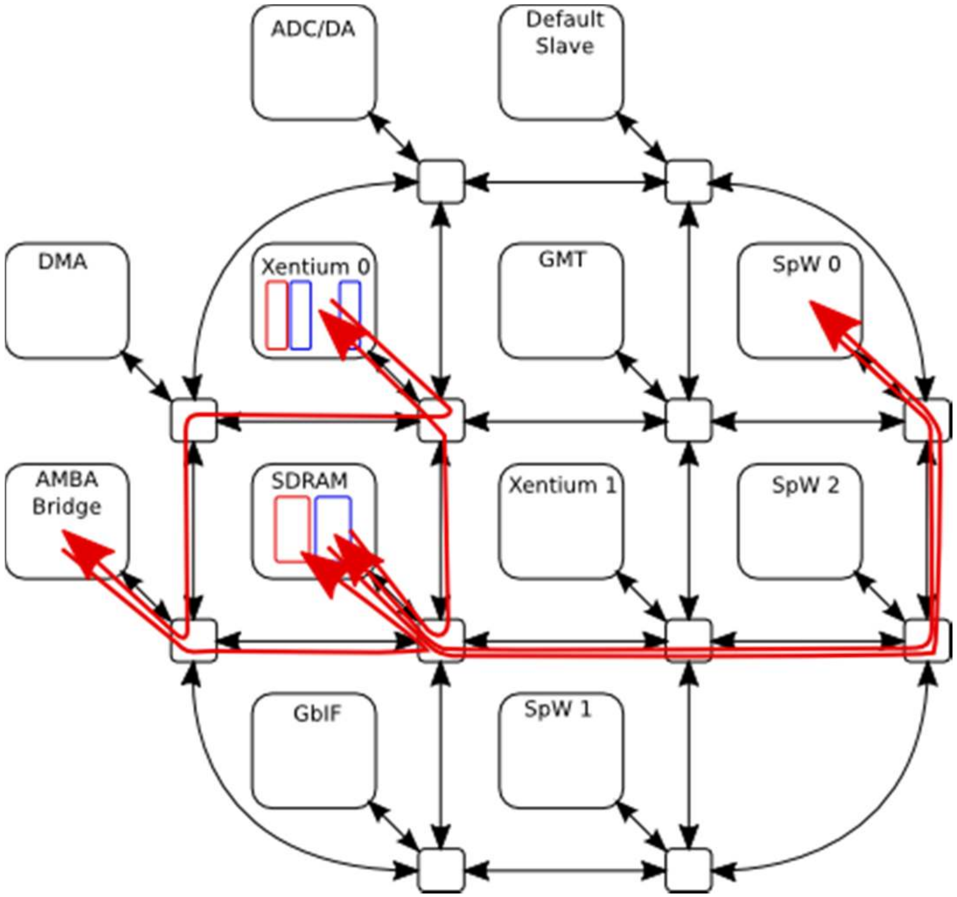
Bit Plane Encoder:
0b0001
0b1000
0b1010
0b1110

MPPB Benchmarking

- Onboard processing Case 1
  - 128 Complex-FIR 80% DDC (10M samples)
  - Compressor (Rice Encoder)

- Benchmark concerns are
  - Speed difference, using more Xentiums makes no sense
  - Encoder is purely functional

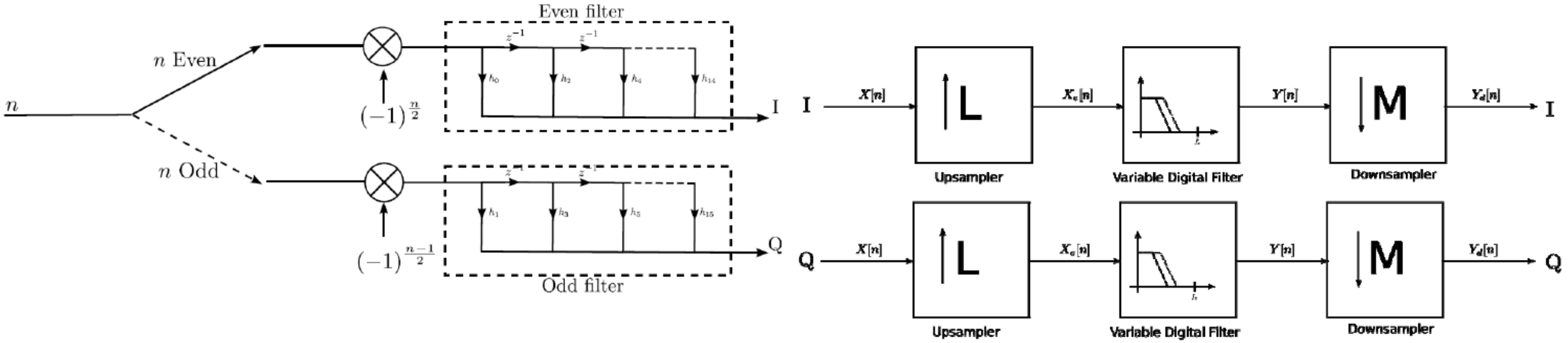
MPPB Benchmarking



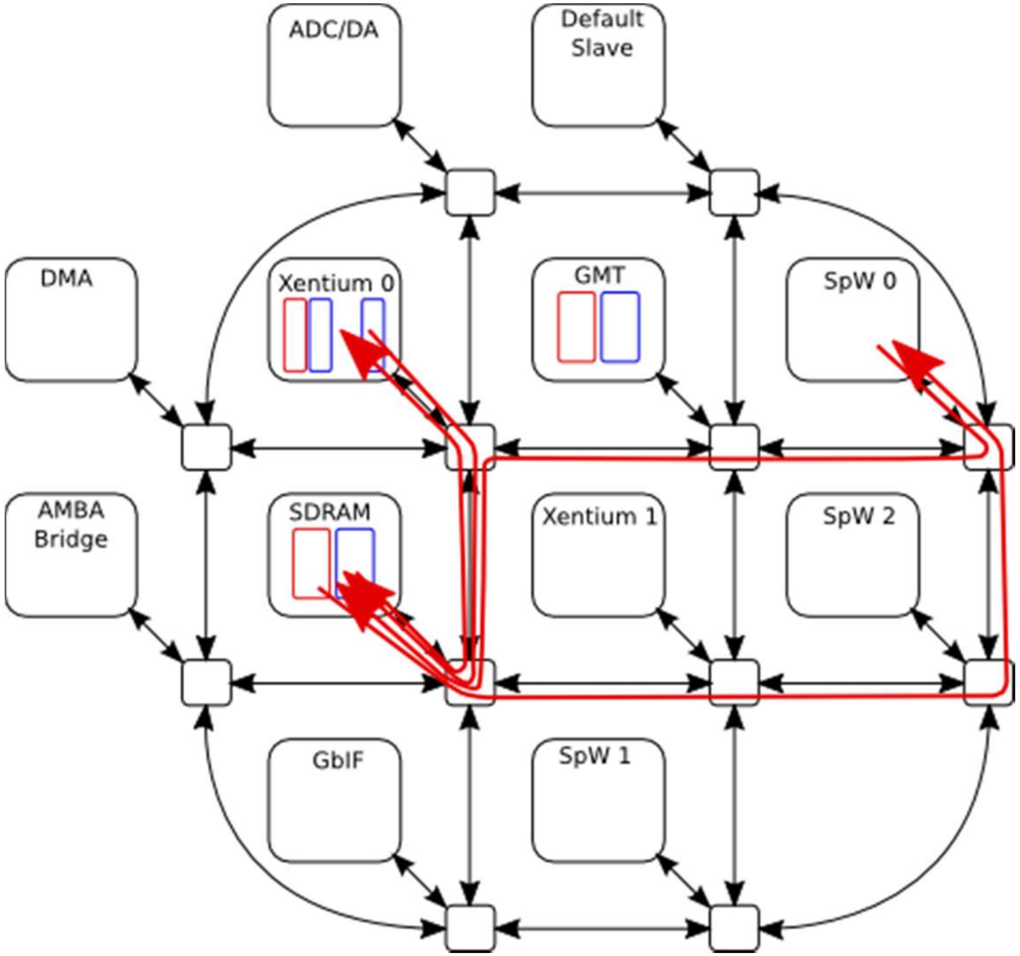
Benchmark 4	Data throughput	Compression factor
Overall	5.8 MS/s	1.1
	Execution time	
Filtering	1.7 s (10000000 samples) 5.8 MS/s	
Decimation		
Compression	732 ms (32768 samples) 44 kS/s	

MPPB Benchmarking

- Onboard processing Case 2
    - Demodulation and 80% Digital Down Conversion



MPPB Benchmarking



Benchmark 5	Maximum throughput [samples (16bit) / sec]	Processing Latency [cycles / ns]
I/Q demodulation only	-	-
Decimation Filter only	-	-
Overall (Demodualtion and decimation)	4.8 MS/s	17246 cycles per 1000 samples

MPPB Benchmark software

- Leon proves to be a bottleneck in the design (interrupts, DMA and compression)
  - Should be dealt with by Xentium itself

- MPPB system itself offers a lot of options
  - e.g. distributed heterogeneous memories, different synchronization mechanisms

- Debug capabilities are essential, e.g. in-system or GDB hook

- No PhD required for software development but experience with VLIW / Embedded systems / Distributed memory systems is highly preferred

MPPB benchmark

- Open for interpretation in some cases
- Would benefit greatly from having reference code, and reference input and output (CCSDS standards not that clear for the average reader)
- Lack of parallelism in benchmark might not show the best of any multi-core (MPPB) system
- Some statistics are a bit peculiar (compression ratio)

Outline

- MPPB platform hardware
- MPPB / Xentium® software development

- MPPB validation & benchmarking
- MPPB FDIR
- ASIC migration & radiation hardening
- MPPB lessons learned / future directions
- MPPB demonstration
- Q&A

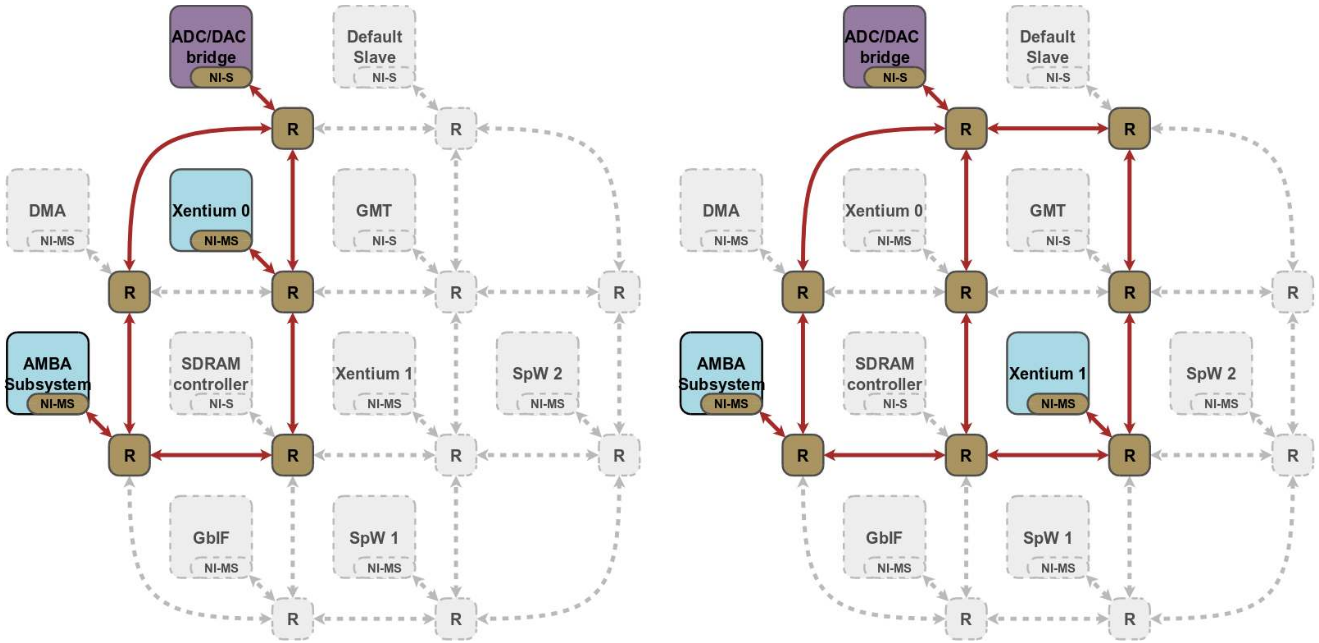
MPPB FDIR Capabilities Assessment and demonstration

- MPPB has many redundant on-chip resources
  - Xentiums
  - Memories – DDR, Memory Tile, Tightly Coupled Memories
  - Digital interfaces – 3 SPW and 1 Gbit interface
  - Debugging – SPW-RMAP and UART-DCOM

- Xentium task migration – Benchmark B2
  - from ADC → Xentium 0 → DAC
  - to ADC → Xentium 1 → DAC

MPPB FDIR Capabilities Reconfiguration

- Proposed Xentium task migration
  - All components are memory mapped
  - Redirection of ADC/DAC input/output
    - Update of Xentium address
    - NoC routes automatically modified



MPPB FDIR Capabilities
Run-time reconfiguration

- Seamless task migration
    - No data loss or corruption
    - Useful for run-time dependability checks

- Demonstration to be implemented – Benchmark B2
    - Control performed by the LEON
        - Prepare X1 to run the same code as X0
        - Update ADC/DAC transfers
        - At the end of a block, start X1 instead of X0

Outline

- MPPB platform hardware
- MPPB / Xentium® software development
- MPPB validation & benchmarking
- MPPB FDIR
- ASIC migration & radiation hardening
- MPPB lessons learned / future directions
- MPPB demonstration
- Q&A

MPPB Radiation Hardening

Inventory and assessment of sensitive elements

- All MPPB IPs are susceptible to radiation effects
  - SEU, SET, SEFI, SEL, TID, ...
- Selection of components to be assessed
  - Xentium
  - Xentium Network Interface
  - Memory Tile (Slave Network Interface + SRAM)
  - NoC routers
  - ADC/DAC interface
  - SPW interface
- Investigation
  - Effects of SEEs on the system for each component

MPPB Radiation Hardening Mitigation techniques

- Three techniques used
  - Process → DARE 180 (Enclosed Layout Transistor) or rad.-hard STM65
  - SRAMs → EDAC
  - System → Watchdog

Mitigation Techniques	Radiation Effects					
	TID	SEL	SEU	MBU	SET	SEFI
Hardening by design	X	X	X		X	
EDAC			X	X		
Watchdog						X

MPPB Radiation Hardening
Hardening impact

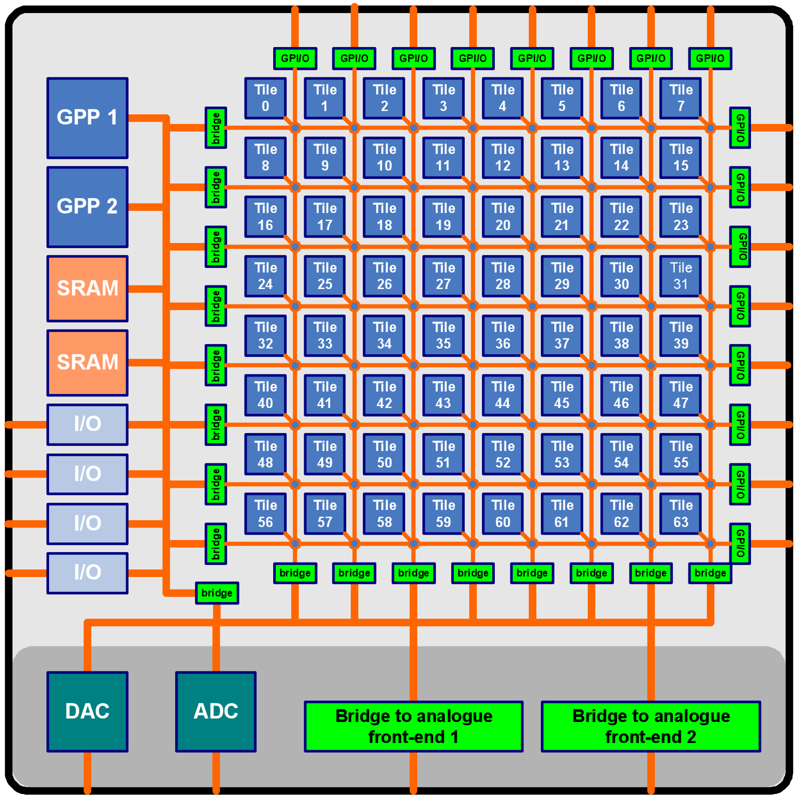
- Expected impact of hardening
  - DARE 180nm
    - Area: 2x-4x increase
    - Power consumption: 2x increase
  - STM 65nm (rad.-hard)
    - *Library not available*
  - EDAC
    - 30-60% increase of SRAM size (SRAM bits + code logic)
    - Timing penalty on SRAM accesses
  - Watchdog
    - Negligible impact

ASIC migration
rad.-hard 65nm *estimations*

- Xentium VLIW DSP core in rad.-hard 65nm CMOS
  - Clock: 300 MHz
  - Performance: 1.2 GMACs/s
  - NoC per link: 9.6 Gbit/s
  - Area: 1.1 mm^2
    - 75% gates utilization
    - Including NoC interface
- Many-core SoC example
  - 48 Xentium processing tiles
  - 16 memory tiles
  - 60 NoC routers
    - 8×8 mesh

→ **60 Giga MAC operations/s**

→ **60 mm^2** (75% gates utilization)



Outline

- MPPB platform hardware
- MPPB / Xentium® software development

- MPPB validation & benchmarking
- MPPB FDIR
- ASIC migration & radiation hardening
- MPPB lessons learned / future directions
- MPPB demonstration
- Q&A

MPPB
Lessons learned

- MPPB architecture
  - For scalability; distribute data flow control in NoC
  - NoC provides high bandwidth
    - No I/O and memory bottlenecks
  - For scalability; avoid central interrupt handling

- MPPB programming aspects
  - Distributed data flow fits nicely with streaming applications
  - For progammability; debugging/tracing capabilities on NoC essential
  - For scalability; multi-core programming SDE essential

- Radiation hardening
  - Technology independent design; design blocks can be mapped on standard process

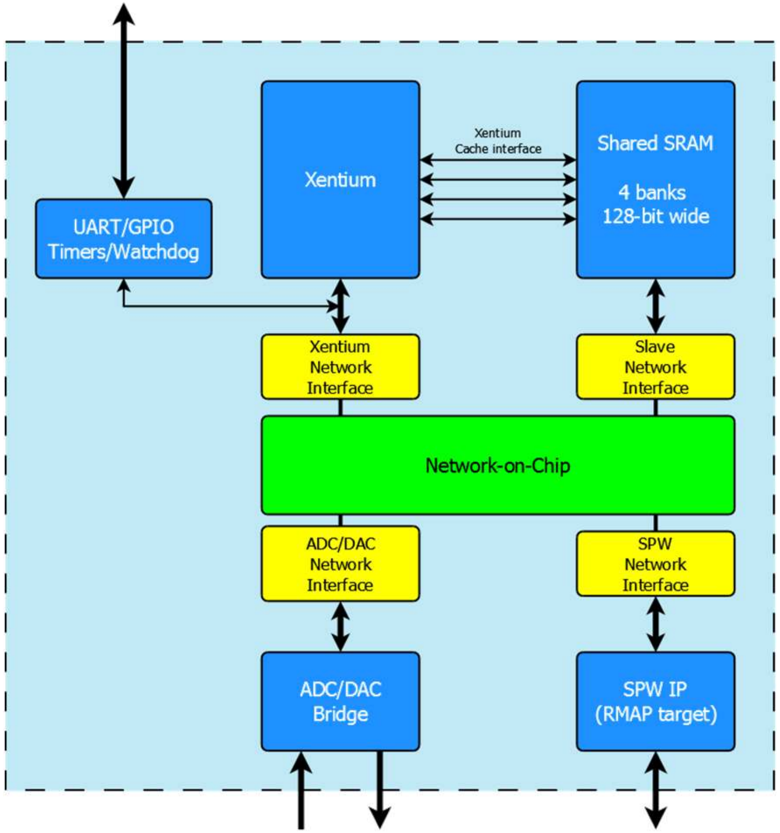
MPPB
Follow-up activities

- ESA TRP activities
  - NGDSP CCN (Astrium)
    - Evaluation study of MPPB system for space applications
  - DARE+ (Imec, Recore Systems)
    - Rad.-hard prototyping of MPPB elements in DARE180 (Jul ’11 – Jun ’13)
- ESA NPI activity
  - Development of methodologies and tools for predictable, real-time LEON/DSP-based embedded systems (2011 – 2013)
    - Performed by Politecnico di Milano (Polimi)
    - Supported by Recore Systems / MPPB

Rad.-hard DSP and NoC prototyping in DARE180

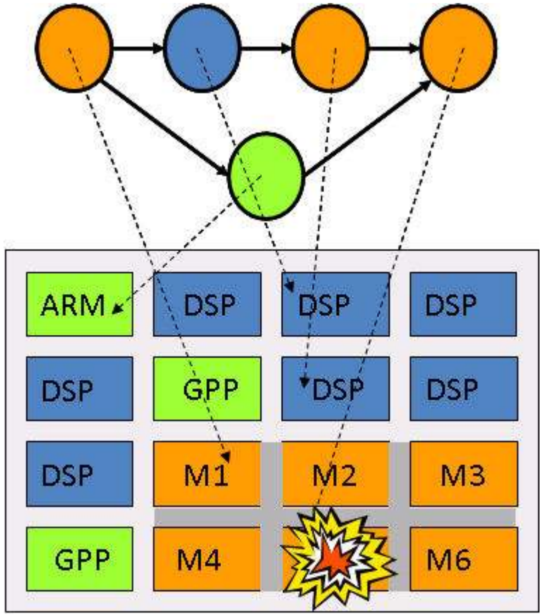
- DARE+
  - Rad.-hard prototyping of MPPB elements in DARE180
  - ESA TRP activity, 2011 – 2013

- ASIC Prototype
  - DARE180 CMOS technology
    - Available area: 5x10 mm^2
  - Architecture
    - 1 Xentium core @ ~100MHz
    - SpW-RMAP interface
      - Connects to external host processor
    - Bridge interface to external ADC/DAC
    - Small memory tile



Related multi-core research

- **CRISP** (FP7)
  - *Dependable & Reconfigurable multi-core SoC*
  - 01-01-’08 / 30-04-’11
  - **www.crisp-project.eu**
- **ALMA** (FP7)
  - *High-level (reconfigurable) multi-core programming and simulation tools*
  - 01-09-’11 / 31-08-’14
  - **www.alma-project.eu**
- **DeSyRe** (FP7)
  - *Fault-tolerant & reliable SoC and NoC*
  - 01-10-’11 / 30-09-’14
  - **www.desyre.eu**



- *Dynamically detect and circumvent faulty hardware*
- *Graceful degradation*
- *Fault-tolerant NoC*
- *Efficient multi-core programming*

Outline

- MPPB platform hardware
- MPPB / Xentium® software development

- MPPB validation & benchmarking
- MPPB FDIR
- ASIC migration & radiation hardening
- MPPB lessons learned / future directions
- MPPB demonstration
- Q&A

Introduction

- Software development overview
- Demo 1:
  - FIR filter example
- Demo 2:
  - Design time reconfigurability
- Demo 3:
  - Introduction to FDIR

Software development

- Application controlled by the GPP (LEON)
  - LEON to peripherals: memory mapped communication
  - Peripherals to LEON: interrupt based notifications
  - Xentium cores used as kernel accelerators (FIR, FFT)

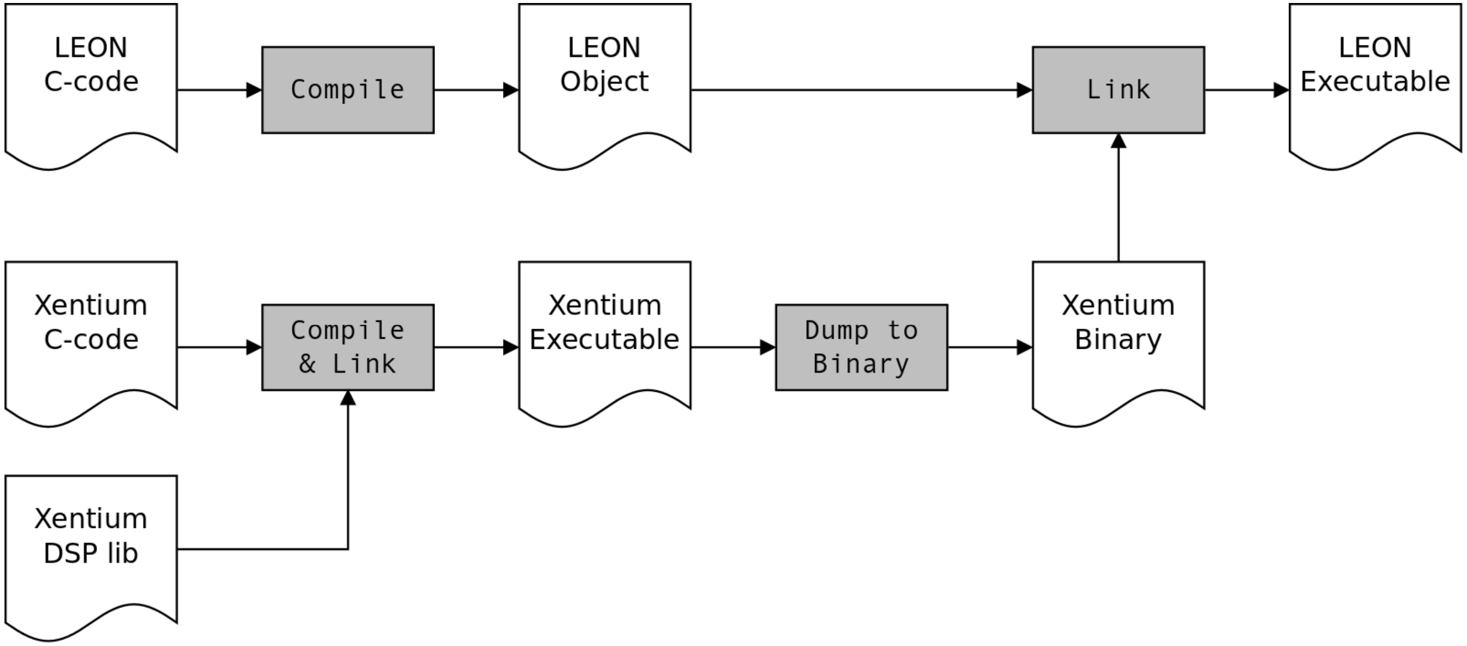
- Software development
  - LEON: C-code
  - Xentium: C-code + DSP kernel library

RECORE

Toolchain

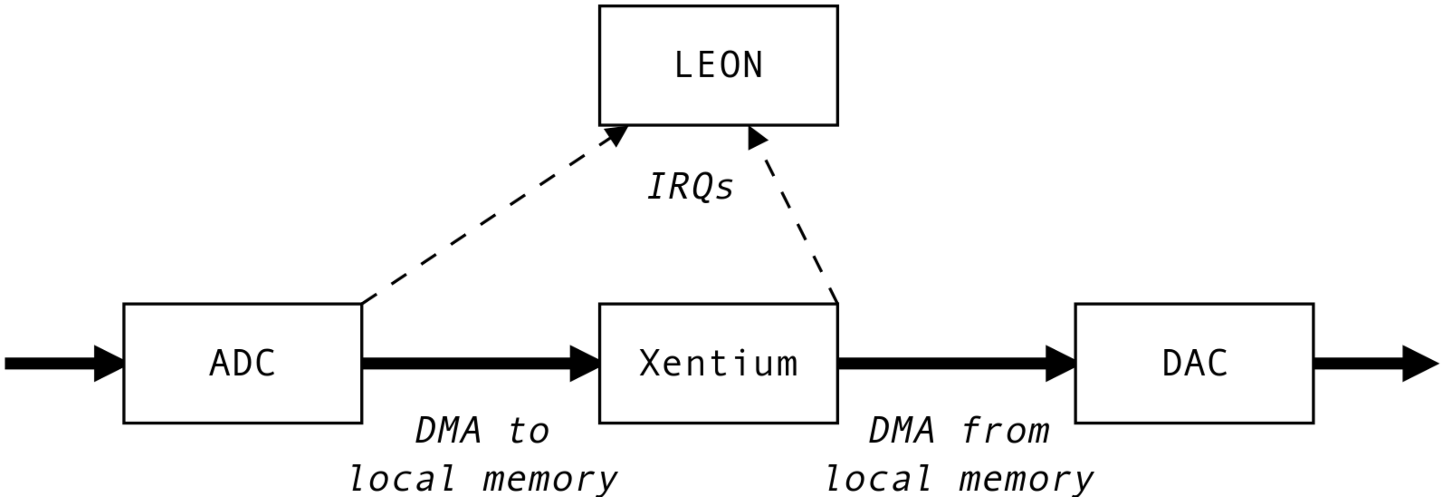
- Xentium binary(ies) linked to LEON executable

- LEON executable uploaded to the platform in SREC format via the UART



Demo 1: Application

- Example: streaming application
  - 16-tap low pass FIR filter
  - ADC → Xentium → DAC
  - Interrupt based synchronization

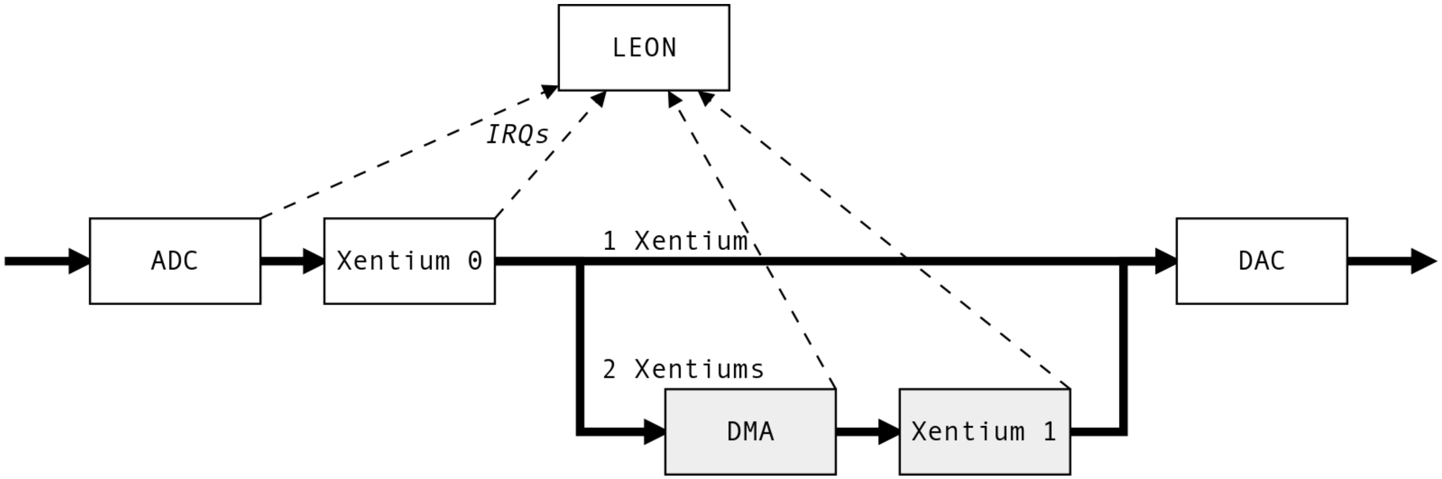


Demo 1: Control

- LEON control
  1) Initialize platform
     1) Configure ADC/DAC
     2) Configure Xentium
     3) Enable interrupts
  2) Start the ADC
  3) Handle incoming interrupts
     - ADC interrupt → Start Xentium
     - Xentium interrupt → Start DAC

Demo 2: Reconfigurability

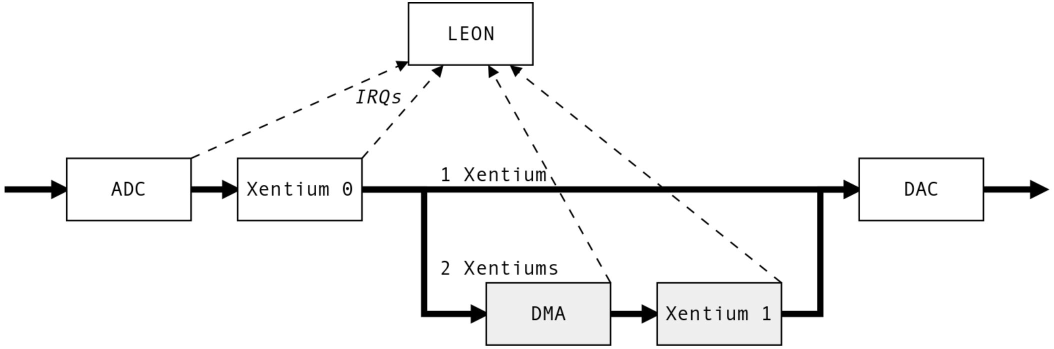
- Demo 1 as baseline
- Design-time reconfigurable application:
  - Use either 1 or 2 Xentiums (FIR filters run in series)



Demo 2: Modifications

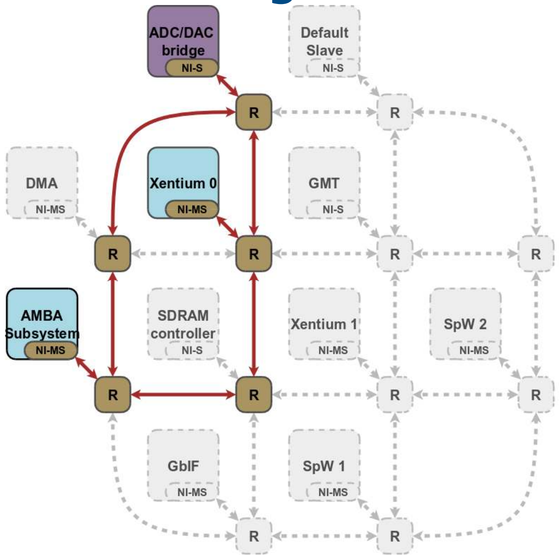
- 2 possible data flows:
  - ADC → Xentium 0 → DAC
  - ADC → Xentium 0 → DMA → Xentium 1 → DAC

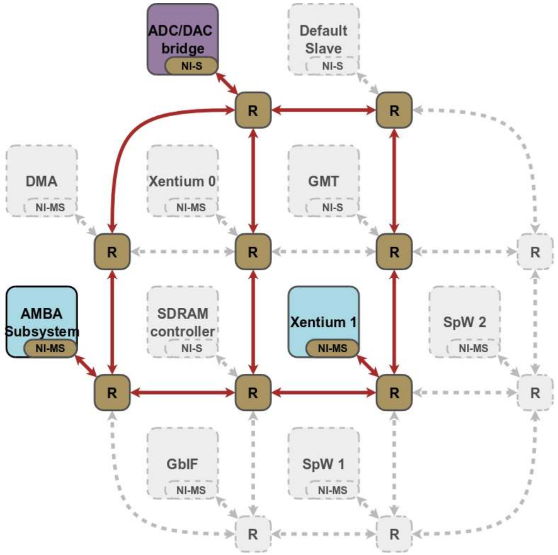
- Reconfiguration:
  - Add DMA and Xentium 1 interrupt routines
  - Start DMA instead of DAC when receiving the interrupt from Xentium 0
  - Modify source of the DAC data transfer



Demo 3: FDIR

- Run-time FDIR with no interruption of service
  - Seamless task migration
  - No data loss or corruption
  - Useful for run-time dependability checks
- Demo:
  - FIR filter running on one Xentium

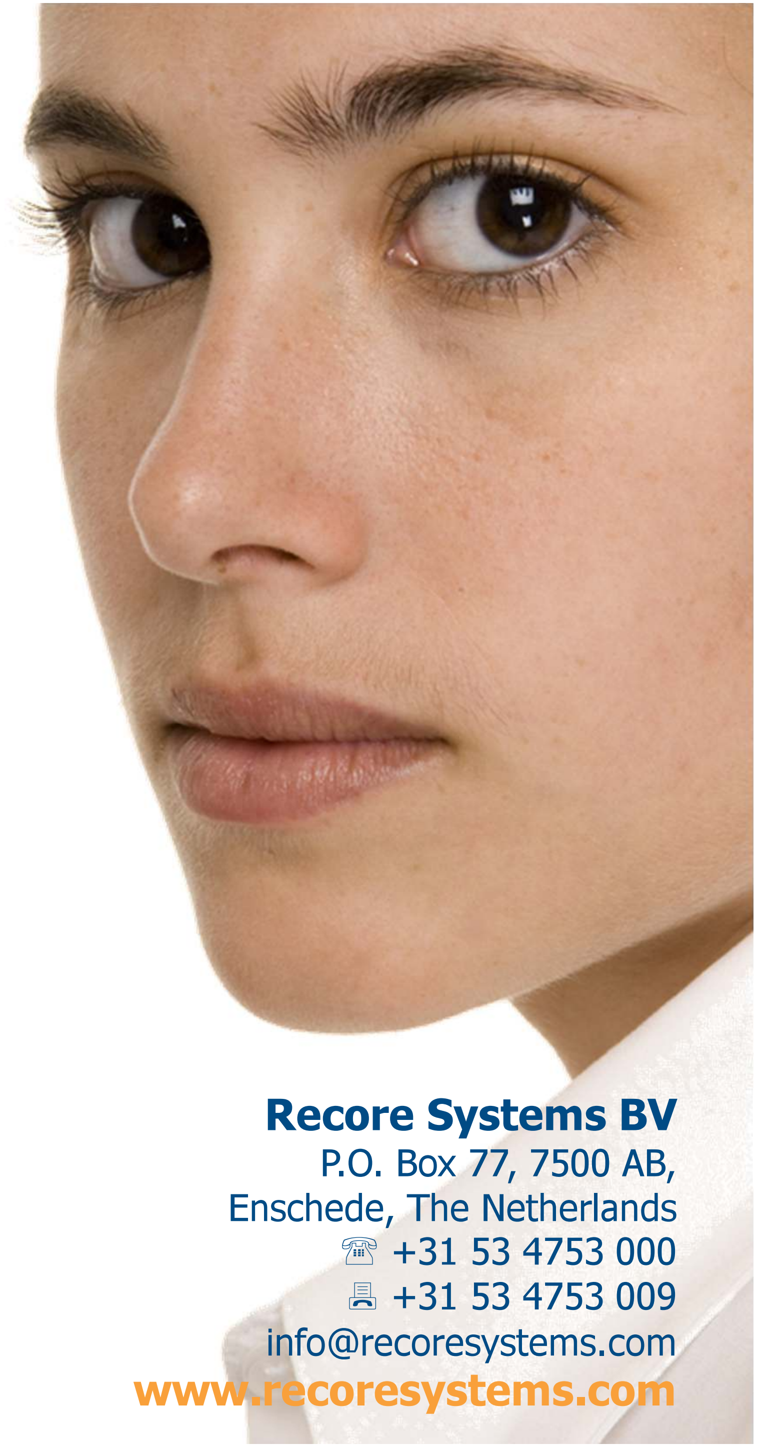




Demo 3: Task migration

- Processing is packet based
  - Seamless migration can be done on a packet boundary
- Pipelined tasks
  - Pipeline integrity must be preserved during the migration
- FIR filters have state (previous input samples)
  - Transfer states between Xentiums
- All components are memory mapped
  - Redirection of ADC/DAC input/output by updating source/destination addresses
  - NoC routes automatically modified

How can we help you?



RECORE®

Recore Systems BV
P.O. Box 77, 7500 AB,
Enschede, The Netherlands
☎ +31 53 4753 000
📠 +31 53 4753 009
info@recoresystems.com
www.recoresystems.com