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

- Running I/O concurrently at \textbf{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

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

- 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

![Diagram showing the process of image data compression with stages like Discrete Wavelet Transform and Bit-Plane Encoder.](Image)
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)
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

<table>
<thead>
<tr>
<th>Benchmark 4</th>
<th>Data throughput</th>
<th>Compression factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall</td>
<td>5.8 MS/s</td>
<td>1.1</td>
</tr>
<tr>
<td>Execution time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filtering</td>
<td>1.7 s (10000000 samples)</td>
<td>5.8 MS/s</td>
</tr>
<tr>
<td>Decimation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compression</td>
<td>732 ms (32768 samples)</td>
<td>44 kS/s</td>
</tr>
</tbody>
</table>
MPPB Benchmarking

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

<table>
<thead>
<tr>
<th>Benchmark 5</th>
<th>Maximum throughput [samples (16bit) / sec]</th>
<th>Processing Latency [cycles / ns]</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/Q demodulation only</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Decimation Filter only</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Overall (Demodulation and decimation)</td>
<td>4.8 MS/s</td>
<td>17246 cycles per 1000 samples</td>
</tr>
</tbody>
</table>
- 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
- **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 SEE on the system for each component
Three techniques used

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

<table>
<thead>
<tr>
<th>Mitigation Techniques</th>
<th>Radiation Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TID</td>
</tr>
<tr>
<td>Hardening by design</td>
<td>X</td>
</tr>
<tr>
<td>EDAC</td>
<td></td>
</tr>
<tr>
<td>Watchdog</td>
<td></td>
</tr>
</tbody>
</table>
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
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 mm²
  - 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 mm² (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 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 programmability; 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
ESATRIP 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)

ESANPI 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 mm²
  - 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](http://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](http://www.alma-project.eu)

- **DeSyRe (FP7)**
  - Fault-tolerant & reliable SoC and NoC
  - 01-10-'11 / 30-09-'14
  - [www.desyre.eu](http://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
- 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
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?