Many-core DSP architectures

Gerard Rauwerda, CTO & co-founder
Gerard.Rauwerda@recoresystems.com

Recore Systems BV
P.O. Box 77, 7500 AB, Enschede, The Netherlands
℡ +31 53 4753 000
fax +31 53 4753 009
info@recoresystems.com
www.recoresystems.com
Dutch fabless semiconductor company
  - Intellectual Property (IP) licensing
  - Embedded many-core sub-system design
  - SoC integration and embedded SW solutions

Focus on professional market
  - Space, defense, security, ...
  - Customized mix of processing, connectivity, and reliability
    - Digital beamforming (e.g. advanced radar systems)
    - On-board payload processing for space applications
    - ...

Recore Systems
Super-exponentially increasing design process complexity
- Silicon complexity increases due to scaling of technology
- System complexity increases due to larger designs
- Software development is very complex and cumbersome

Overall design technology challenges
- Design productivity & reuse productivity
- Power consumption
- Manufacturability
- Reliability
Complex systems nowadays require the integration of many processors of various types (MCU, GPU, DSP, ...)

Software development on these heterogeneous many-core systems is very complex and cumbersome.
Reconfigurable functionality of a SoC

Based on ITRS 2009
**General Stream Processor**
- Heterogeneous many-core SoC
- General Purpose Processor
- 45 Xentium DSP cores
- 10 smart memory tiles
- Network-on-Chip (NoC)

**Reliable processor for space**
- Scalable DSP subsystem (new)
  - 2 Xentiums
  - Network-on-Chip
- GPP subsystem (existing)
- NoC bridge
Xentium®: Powerful DSP

- Suitable for scalable many-core SoC design or as accelerator core
  - Competitive combination of
    - Small silicon footprint,
      - computational power and low power consumption
  - Customizable core
  - Predictable and deterministic behaviour

- Straightforward Xentium integration for customized SoC
  - Optimize memory size (small, large)
  - Choose interfaces
  - Optimize # of accelerators
Xentium Architecture - Highlights

- Programmable high-performance DSP
  - High instruction-level parallelism
- Data precision
  - 32/40-bit fixed-point
  - 16-bit SIMD
- Features
  - Single-cycle latency Data Memory
  - Single-cycle instruction cache latency
  - Short 3-cycle pipeline
  - Efficient complex MAC execution
  - Register bypassing (latency, energy efficiency, code size)
  - Loop buffer (energy efficiency, code size)

<table>
<thead>
<tr>
<th>CMOS</th>
<th>GMAC/s</th>
<th>Clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>65 nm</td>
<td>1.6 GMAC/s</td>
<td>400 MHz</td>
</tr>
<tr>
<td>90 nm</td>
<td>0.88 GMAC/s</td>
<td>220 MHz</td>
</tr>
</tbody>
</table>
Xentium - datapath
parallel execution units

5 register files
- RA: 16x32
- RB: 8x32
- RE: ...

2 data load/store units
- E0: 32, 64
- E1: ...

2 ALU units
- A0
- A1

1 ALU with control & pack
- C0

2 multiply units
- M0
- M1

1 ALU with pack
- P0

2 ALU w/ shift operations units
- S0
- S1
Xentium Tool chain

- Xentium C compiler
  - ANSI/ISO-standard C
  - Built-in functions for Xentium specific operations
  - Mix C and assembly functions calls

- Xentium assembler
  - Clean and readable
  - Extensive built-in preprocessor
  - Standard assembler directives

- Compile, assemble & link a program in a single step

- Xentium instruction set simulator
  - Trace program execution
  - Program execution cycle count

- Runs on 32- and 64-bit Linux
Xentium Eclipse Plug-in

Overview

- Use features provided by the Eclipse IDE for C/C++
- Integrates the command line Xentium tool chain
- Diagnostics support for Xentium compiler
Tomorrow’s General Stream Processor

- A radically **scalable many-core** architecture
- Run-time resource management for **dynamically reconfiguring** cores and network-on-chip
- **On-chip dependability** infrastructure for hardware test, diagnose and repair
- **Low-cost** and super **high-performance** application demonstrators
- **Prototype** includes **45** Xentium® DSP cores
The GSP X-45 prototype (includes 45 Xentium cores)
The GSP X-45 prototype PCB
- Dynamically determine the assignment of streaming application kernels to available resources
  - Configure tile processors
  - Configure Network-on-Chip
- Objectives
  - Dramatically improve the utilization of available cores
  - Improve dependability by dynamically circumventing faulty hardware
  - Simplify compilers by separating communication from computation

Run-time mapping

A streaming application is modeled as a series of communicating parallel processes to be applied for each element in a stream of data
Hide (software) complexity

- Dynamically determine mapping
- Automatically reconfigure
  - Processors / Network-on-Chip
  - Adapt to platform changes
- Running on control processor
- Managing HW resources in many-core system
- Remapping on faults, anticipating on power
Run-time resource management

- Requires annotated task graphs
  - Specifying process and communication
- Maps
  - tasks to processing cores
  - inter-task communication to network resources
- Heterogeneous platform resources
  - Cores differ on access to memory and IO
- Adding and removing applications changes platform
  - Reconfiguring system resources
- Construction of annotated task graphs
  - Fairly straightforward for applications with natural task-level parallelism
    - Radar Beamforming
  - Difficult in general
Run-time reliability management

- Reduce Cost for Fault Tolerance
  - Small Fault Free area manages unreliable resources
  - Fault Prone resources for flexibility, reconfiguration

- System reliability managed at runtime:
  - Application requirements
  - System constraints
  - Fault types and density

![Diagram showing task list and fault-free area management](image-url)
Hide (hardware) complexity

Keep it Simple for the Programmer

- Platform-independent application code performance estimation and optimization
- Identification of possible partitions and placing & routing on different underlying architectures
Fields of expertise

- Heterogeneous many-core architecture design
  - On-chip interconnect (NoC) IP & Processor IP (Xentium DSP)
- Programming of heterogeneous many-core systems
  - Run time mapping within a heterogeneous many-core system
  - Parallel programming models
- System analysis, generation, simulation and verification tools

Multi-core operating system with runtime resource scheduling

- Customer’s software application (in C)
- On-chip interconnect
- Xentium DSP
- Processor / Memory

Definition of desired behaviour (what)
Management of hardware resources (how)
Data exchange and on-chip communication
Hardware blocks selected to efficiently perform the application’s tasks
FP7 European Research Projects

- **ALMA**
  - ALgorithim parallelization for Multicore Architectures [2011-2014]

- **DeSyRe**
  - on-Demand System Reliability [2011-2014]

- **Sensation**

- **Polca**
  - Programming Large Scale Heterogeneous Infrastructures [2013–2016]