Fault Detection and Recovery in a Teradevice Dataflow Multicore System

Theo Ungerer
Chair of Systems and Networking
University of Augsburg, Germany

MEDIAN Workshop, 31 May 2012
Outline

- Future Many-Cores
- Reliability Issues in Future Many-Cores
- Fault Detection in Future Many-Cores
- Teraflux project approach
- Conclusion
Future Many-Cores

- Moore’s law:
  Doubling the number of transistors per processor die every 24 months

- „Technology scaling will continue for at least another 10 years“
  (Intel keynote speaker at HiPEAC 2011)
Future Many-Cores

- "In a couple of years we will have 50 billion transistors on a chip."
  (Kathryn O’Brien, IBM)
  - Pentium4 host 40 million transistors
  - Space for >1K Pentium4 cores!

- "Integrated circuits and tightly coupled systems will integrate up to 1000 billion devices by the year 2020."
  (EC Objective ICT-2009.8.1: FET proactive 1: Concurrent Tera-device Computing)
  - Space for >250K Pentium4 cores!

Is this realistic?

What about the
- parallelism wall,
- costs,
- time to market,
- technology problems, and
- reliability?
Reliability Issues in Future Many-Cores

- Common practice:
  - Chips are and will be delivered with faulty devices (permanent faults)
  - Intel Pentium II → Celeron (Defect Cache Areas)

- Number of soft errors will increase

- Chips will age

Chips will age

Wearout effects

Sources: Patrick-Emil Zörner (Wikipedia)
Systems will rely on unreliable components
Objective
- Detection of faulty elements and self-healing in future many-cores
- i.e. processors with 1000 and more cores

Our Idea for fault detection:
- Some cores on a many-core can be dedicated to fault detection and reliability maintenance
- **TERAFLUX – EC-FP7-Project**
Many-Core with Holes

Fault-free many-core

Faulty many-core
  • faulty cores
  • faulty links
Fault Detection and Recovery

- **Target**
  - Keep functionality even when components fail
  - Graceful degradation by using less cores

- **Proceeding**
  - Clustering of monitored elements
  - Monitoring and fault detection by Fault Detection Units
  - Recovery and restart of tasks by scheduler
Fault Detection Units (FDUs)

- Clustering of cores to FDU
  - HW-fixed
  - Dynamic

- Receives heartbeats
- Assesses core states
- Assesses cluster states
- Notifies scheduler
- Reconfigures cluster
Faults at NoC Level

- OS and Node Manager preserve the **illusion of a regular mesh** for applications
- Bypassing faulty elements
  - partially adaptive routing
  - incorporating a turn-model to guarantee deadlock freedom
- FDU placement in a mesh-structured NoC
  - Core to FDU assignment → Clustering
  - Prevent heartbeat bottlenecks
  - Prevent heartbeat collisions due to faults
FDU Internal Behaviour

Behavior derived from Autonomic/Organic Computing

- **Monitors** managed elements
- **Analyses** received information
- **Plans** long term decisions
- **Executes** decisions to maintain the cluster reliability
- **Communicates** with the Scheduler
Part of EC Project TERAFLUX

- Targets many-core with 1000 cores
- Threaded dataflow combined with transactional memory
- FDUs for fault detection
Architectural Template of a TERAFLUX system
Fault Detection Unit (FDU) Concept Overview

Node/Cluster

Node-Manager

D-TSU

D-FDU

Task placement

Monitoring

CORE

CORE

CORE

CORE

Neighbouring D-FDU

Neighbouring D-FDU

Neighbouring D-FDU
Fault Detection at Core Level

- L-FDU provides reliability related information to D-FDU via alert/heartbeat messages

- **Soft Error Detection on core-level**
  - Control flow error checking techniques for dataflow threads with control flow instructions
  - Double execution for dataflow threads
  - Recovery by thread-restart mechanism
  - Memories protected by ECC
Control Flow Error Checking

- Code instrumentation on basic block level:

- Detection of control flow errors in DF-thread execution:
  - Temporal control flow monitoring
    - DF-Threads pre-fetch their data → no interrupts during thread execution
    - Exploiting the thread execution time predictability
  - Logical control flow monitoring
    - If threads contain control flow instructions
    - Detection of wrong control flow behavior (illegal jumps, ... )

- L-FDU alerts D-FDU in case of time-out or wrong control flow.

- Dataflow thread execution commitment is deferred until all checks are done.
Control Flow Error Checking

- Evaluation on an in-house SystemC model of an embedded RISC processor

- Preliminary fault injection studies show:
  - More than 30% of bit flips in the instruction memory can be detected
  - Number of endless loops is reduced by more than 75% compared to executions without check unit

- Required overhead (on average):
  12.2% additional execution time,
  15.0% increased code size
Double Execution

- Detects control flow AND data errors

- Each execution generates signature of output results.
  - At completion compare the two signatures, if consistent, the D-TSU writes its results to subsequent thread frames.
  - If not, no commitment and recovery.
Recovery

- The recovery mechanism exploits the data-flow execution model, which permits recovery on thread granularity.

- D-TSU buffers writes until the D-FDU gives feedback
  - In case of a fault a thread is rescheduled
  - In case of a fault free execution the buffered writes are committed to main memory

- WIP: incorporate the transactional memory mechanism in the recovery mechanism
Conclusions

- Reliability issues in future many-cores lead to the quest to "build reliable systems from unreliable components"

- Core-router-link level: extended FDUs for fault detection

Some open problems:
- Recovery (eased by dataflow threads)
- Fault tolerance on a many-core
- Hard real-time and unreliable components
Any Questions?

Theo Ungerer
ungerer@informatik.uni-augsburg.de

TERAFLUX Team UAU:
Bernhard Fechner
Sebastian Weis
Arne Garbade
Julian Wolf