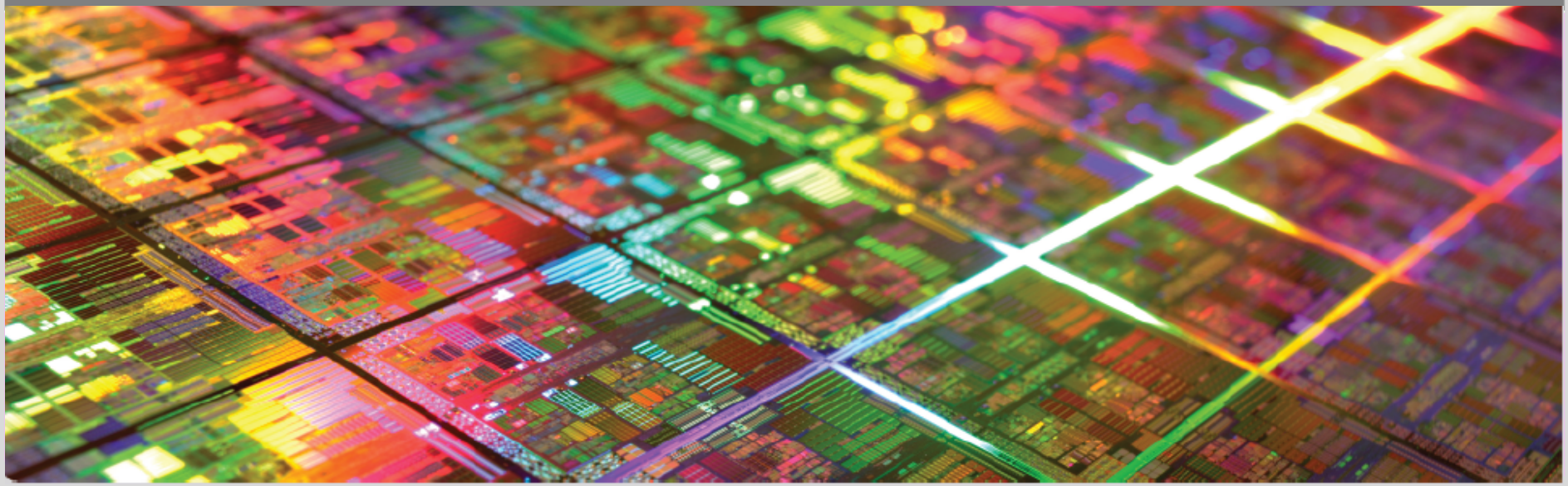
Karlsruher Institut für Technologie

WG4: Language Integration & Tools

Wolfgang Karl

Fakultät für Informatik – Lehrstuhl für Rechnerarchitektur und Parallelverarbeitung



Agenda

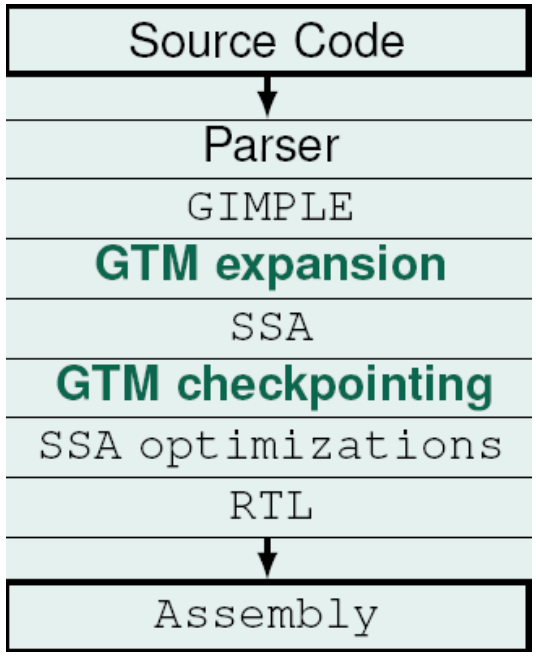
- Goals of WG4 Language Integration and Tools
- Overview of language integration efforts
- Scheduled Talks

Goals of WG4: Language Integration and Tools

- Coordinate European research activities:
  - Integration of TM supports within mainstream languages and development frameworks
    - extensions to compilers
    - virtual machines for languages relying on managed runtime systems
    - Explore TM integration in Message passing-oriented or data parallel languages
  - Boost standardization process for APIs
  - Supporting tools: debugger, profiler, performance analyzer and performance tuning
- Enable and support collaborations
- Platform for researchers on these topics

Language Integration

- TM support for **C in GCC**
  - GCC's intermediate representations with new GTM passes.
  - Joint project of KIT (Prof. Karl, M. Schindewolf) and INRIA (Prof. Cohen)
  - Now: revised and maintained by Red Hat Inc.



Language Integration efforts

- **Languages C and C++**
  - Many different STM implementations available
  - Standardization Initiative led by Intel
    - TM language constructs for C++ proposed by People from Intel, IBM, and Oracle
      - Keyword `__transaction [[attribute]]{}`
      - Attributes define the semantic of a Txn
    - Intel published ABI for compatibility at assembly-level
      - STM function names
      - Calling conventions
      - Transaction nesting
      - Barrier optimizations
      - Irrevocability

Language Integration efforts

- **Languages that would benefit from further standardization**
  - Java,
  - C#,
  - Haskell,
  - Phython,
  - OCaml,
  - Clojure,
  - Lisp,
  - Perl,
  - Smalltalk,
  - Fortress,
  - ...

1st Plenary Meeting – WG4

- **Session 1: Language Support for TM**
- Daniel Goodman: **MUTS: Native Scala Constructs for Software Transactional Memory**
  - Keyword-based TM extension of Scala
  - Enhance parser with TM capabilities
- David Kitchin: **Atomicity and Structured Concurrency**
  - Orc programming language with atomic contruct
  - Formalizes notion of atomicity
- Ismail Kuru: **Integration of transactional memory support into a data-flow extension of OpenMP**
  - TERAFLUX: Combine data-flow and TM
  - Prototypical implementation in GCC (extend OpenMP and combine with TM branch)
  - Define semantics of transactions nested within OpenMP tasks

1st Plenary Meeting – WG4

- **Session 2: TM programmability support and tools**
- Jeremy Singer: **Programmer and Runtime Policies for Adaptive Many-Core Parallelism**
  - Novel Java extension
  - Encodes accuracy of program modules
  - Adaptively spawns threads
- Martin Schindewolf: **Methods and Strategies to Rate and Optimize Transactional Memory Applications**
  - TM-Opt: Optimization of TM applications
  - Close gap between low-level TM research and programmer's view
  - Provide tools to guide programmer

1st Plenary Meeting – WG4

- **Session 2: TM programmability support and tools**
- Guy Korlan: **Crafting a High-Performance Ready-to-Go STM**
  - Deuce: efficient full-featured Java STM framework
  - No compiler or JVM modifications required
- Joao Cachopo: **JVSTM and its applications**
  - Java library implementing a multi-version STM
  - Applied to several domains and used productively

Discussion

- **Language integration**
  - Are standardization activities useful?
    - Does a common API help?
    - Industry activities – cannot be ignored!
- **Runtime systems**
  - Adaptation
- **Tools**
  - **What are the features?**
    - Debugger
    - Profiler
    - Performance analysis / tuning