Methods for predictAble Design of heterogeneous Embedded Systems with adaptivity and reliability Support

MADNESS will focus on improving Embedded Systems design predictability, considering new features, such as adaptivity and fault tolerance.

The main goal of the project is to define innovative methodologies for system-level design, able to guide designers and researchers to the optimal composition of embedded MPSoC architectures, according to the requirements and the features of a given target application field. The proposed approach will tackle the new challenges, related to both architecture and design methodologies, arising with the technology scaling, the system reliability and the ever-growing computational needs of modern applications.

The proposed methodologies will extend the classic concept of design space exploration to:

• improve design predictability, bridging the so called "implementation gap", i.e. the gap between the results that can be predicted during the system-level design phase and those eventually obtained after the silicon implementation;

• consider, in addition to traditional metrics (such as cost, performance and power consumption), continued availability of service, taking into account fault resilience as one of the optimization factors to be satisfied;

• support adaptive runtime management of the architecture, considering, while tailoring the architecture, new metrics posited by novel dynamic strategies and advanced support for communication issues that will be defined.

Project Coordinator
Name: Luigi Raffo
Institution: University of Cagliari (Italy)
Email: raffo@unica.it
Project website: http://www.madnessproject.org

Duration: 36 months
Start: 2010.01.01
Total Cost: 2.92 ME
EC Contribution: 1.95 ME
Contract Number: INF50-ICT-248424

Partners:
- Università degli Studi di Cagliari (Italy)
- Università della Svizzera Italiana (Switzerland)
- Silicon Hive BV (The Netherlands)
- Lantiq Deutschland GmbH (Germany)
- Universiteit van Amsterdam (The Netherlands)
- Universiteit Leiden (The Netherlands)
- DAEDALUS Informatics Centrum Dortmund (Germany).

---

Support for heterogeneity at compilation level. Configurable compiler architecture.
- Heterogeneous system features, different compiler to be hardened in one single architecture.
- Configurable compiler or group of compilers.
- Extension to Linux Compiler Stack to create a new language (VASSO), with X86 additional data types and function calling.

---

Support for heterogeneity at compilation level: Instruction Set and Entry Layer
- Management of system development.
- (Software or Hardened) Configuration, adaptation and evolution to the silicon target in the future.
- A. To help in entry, to run on a local host.
- Target adaptation: on a platform of the firmware components without modification of the IC.
- Standard
- User defined periphery
- Autonomic
- Non-optimal knowledge memory.

NOMAD middlewares components:
- Standard middleware except for the isochronous tuning, scheduling, timing planning and for the communication among units.
- Task adaptation systems, e.g.
  - fault-tolerant (optional management)
  - run, run, hard configuration,
  - self configuration: a new task.

---

Task scheduling approach based on a deadline scheduling when a task blocks on read/write, the processor switches to another task or can very small, slight other class approach.

Scenarios:
1. On FPGA execution and performance evaluation
   - FPGA Based Evaluation platform

2. Application (ALPS) target
   - ALPS operating system
   - Local Operating System
   - Processor

---

Initial implementations of ALPS platform and DAEDALUS framework.

---

Hardware/Software partitioning area: Application (no hard)

---

Task scheduling approach based on a deadline scheduling when a task blocks on read/write, the processor switches to another task or can very small, slight other class approach.

---

Hardware/Software partitioning area: Execution (no hard)

---

Hardware/Software partitioning area: Design (no hard)

---

Hardware/Software partitioning area: Configuration (no hard)

---

Hardware/Software partitioning area: Command (no hard)

---

Hardware/Software partitioning area: Control (no hard)

---

Hardware/Software partitioning area: Process (no hard)

---

Hardware/Software partitioning area: Task (no hard)

---

Hardware/Software partitioning area: Algorithm (no hard)

---

Hardware/Software partitioning area: Dynamic (no hard)

---

Hardware/Software partitioning area: Static (no hard)

---

Hardware/Software partitioning area: Storage (no hard)

---

Hardware/Software partitioning area: Body (no hard)

---

Hardware/Software partitioning area: Soul (no hard)

---

Hardware/Software partitioning area: Heart (no hard)

---

Hardware/Software partitioning area: Brain (no hard)

---

Hardware/Software partitioning area: Body (no hard)