Towards a Formal Software Design Methodology for Predictable Embedded Multiprocessor Applications

Ingo Sander
KTH Royal Institute of Technology
Stockholm, Sweden
ingo@kth.se

Correct-by-Construction Design
The Dream...

- Timing, Throughput
- Cost
- Energy and Power
- Size
- ...

Function Specification

Capture functionality
Executable
Includes environment

Design Constraints

- HW Architecture
- SW Architecture
- Flexible or Fixed

Correct Implementation
- Correct functionality
- All non-functional constraints are fulfilled

Design Process

Platform Specification

No additional verification needed!
Embedded Real-Time Design

Current Situation

- It is very difficult to accurately estimate the performance of an embedded real-time system
- Huge difference between average and worst case execution time

As a consequence

- New designs are rather based on old experiences than on performance analysis
- Sufficient safety margins in form of more powerful components and extra communication bandwidth
- Verification costs are extremely high!

Surely, there must be a better way to design systems...

Why is it so difficult to estimate software?

- Advanced processors are difficult to predict due to mechanisms, that are aimed for improvement of average case performance
- Multiprocessor architectures
  - often share main memory, communication channel and other resources
  - communication time may also be location dependent
Trade-Off: Average Case Performance vs Predictability

Difficult to predict performance
- Difficult to predict memory access time
- Execution time of Pgm1 depends on Pgm2 due to shared resources!

Easy to predict performance
- Predictable memory access time
- Dedicated resources give composability!

Learn from Hardware Design
- Hardware design is predictable
  - Based on simple communication mechanism (clock)
  - Formal models are available (Boolean algebra and synchronous model)
  - Predictable architecture (gate delays)
- Very efficient tools!
  - Timing predictable in range of nano- or pico seconds

Synchronous languages (Esterel, Lustre) have been very successful for safety-critical applications!
What is needed?

- Formal **analyzable models** of the application
- Platforms that can provide **service guarantees**
  - Ideally: totally predictable platforms
  - At minimum: typical values on performance
- **Analysis methods** enabling design of tools
- An **industrial entry language** enabling
  - Simulation of the application model
  - Automatic extraction of analyzable models

---

Models of Computation (MoC)

- MoC specifies semantics of computation and communication
- Abstracts from design language
- Enables application of analysis methods
  - Design space exploration
  - Performance analysis
  - Verification
  - Synthesis
- Active research area
  - Development of new techniques and tools
Motivational example
Streaming application

- If designers are not aware of MoCs, similar applications will likely be implemented with
  - RTOS to schedule the different actors
  - Buffers for the communication between actors
  - Safety margin for buffers

Ingo Sander
Embedded Conference Scandinavia, Nov 5, 2013

Motivational example
Streaming application

- If designer is aware of synchronous data flow (SDF) MoC [Lee1987]
  - Static schedule can be derived (AABCAABCC)
  - Minimal buffer size can be calculated (2+4=6)

Efficient implementation

... but often designers are not aware of MoC theory...

Ingo Sander
Embedded Conference Scandinavia, Nov 5, 2013
ForSyDe (Formal System Design) System Model

- A system is modeled as hierarchical concurrent process model
- Processes of different models of computation (MoC) communicate via domain interfaces
- ForSyDe libraries support the designer in the development of an executable formal model

SystemC-ForSyDe

- SystemC libraries force the designer to develop a structured model enabling formal analysis
- SystemC-wrappers are used to
  - integrate existing models written in other languages
  - co-simulate low-level models running on executable platforms with high-level models
Exporting from ForSyDe-SystemC

Ingo Sander
Embedded Conference Scandinavia, Nov 5, 2013

Overview
Software Synthesis Design Flow

Ingo Sander
Embedded Conference Scandinavia, Nov 5, 2013
Overview
ForSyDe Software Synthesis Flow

- Designer models Executable SystemC Model
- Extraction of Analyzable Model (XML+C)
- Design Space Exploration ➔ Efficient Mapping
- Code Generation for Individual Processors

Ingo Sander
Embedded Conference Scandinavia, Nov 5, 2013

Design Space Exploration

- **Design Space Exploration Problem**: How to map and schedule application processes onto platform resources (processors, memories, interconnect) while satisfying performance and other non-functional constraints processes shall be mapped to which processors?
- **Design constraints**: guaranteed throughput, memory constraints, power constraints, ...
- Problem is formulated as constraint satisfaction problem using constraint programming

Ingo Sander
Embedded Conference Scandinavia, Nov 5, 2013
Synthesis to Target Platform

Process Network Synthesis
1. Implementation of static schedule
2. Generation of internal FIFO buffers
3. Code generation for each process, including function calls for message passing communication

Prerequisites
- Platform provides communication primitives for message passing and service guarantees

Current State
- ForSyDe Haskell and SystemC libraries for four MoCs are publicly available
- Abstract XML-model and C-functions can be extracted from SystemC ForSyDe
- Integration of foreign models using ForSyDe wrappers
  - Co-simulation of legacy code with ForSyDe model
  - Refinement-by-Replacement (Hardware in the Loop)
- Haskell ForSyDe
  - Hardware synthesis backend (ForSyDe -> synthesizable VHDL)
Ongoing Work

- Development of automated design flow
  - Design space exploration method based on constraint programming
  - Software synthesis of ForSyDe system models to an FPGA-based NoC developed at KTH (Johnny Öberg)
  - Development of GPGPU backend
- ForSyDe has is/has been used in the European projects ANDRES, SYSMODEL, iFEST and CONTREX

Thanks for your attention!

ForSyDe WWW: [https://forsyde.ict.kth.se](https://forsyde.ict.kth.se)