FREACSIM - A Framework for Creating and Simulating Real-Time Capable Network on Chip Systems and Applications

Dominik Schoenwetter, Ronald Veldema and Dietmar Fey

Athens, August 26, 2015

Chair of Computer Science 3 (Computer Architecture)
Friedrich-Alexander-University Erlangen-Nürnberg
Source: produkte.bosch-mobility-solutions.de
Source: produkte.bosch-mobility-solutions.de
Outline

1. The Framework FREACSIM
2. Measurement and Results
3. Conclusion
4. Outlook
FREACSIM - Framework for Real-time capable Embedded system and ArChitecture SIMulation

- Is able to generate simulation models of various real-time capable embedded Network on Chip/network interconnected architectures
- Simulation of these models is possible
- For each of the various hardware architectures, a real-time capable, and software-based, routing library with API can be generated
- Real-time capability is achieved by using the real-time operating system eCos
Overview

User Input

noc-generator

XML File (Hardware Description)

*.xml

Hardware Information Header File

*.h

xml-to-sim-model

Hardware Simulation Model

*.h *.cpp

Executable Simulation Model

*.cpp

Application(s) to run

OVP

Run Simulation

Results

Routing Library and Header Files

*.a *.h

used by

Application(s) to run

Run Simulation

Results
noc-generator

- Generates the hardware design (2d-torus, grid, ring or star)
- Generates an xml hardware description file
- Generates a hardware information header file
A Simple $2 \times 2$ Grid Hardware Design

- **C**: Computation node
- **M**: Shared memory
- **R**: Routing node
- **Bidirectional link** (read/write transfers)
Example for a $4 \times 4$ Torus-2D Hardware Design
Possible Settings for Hardware Generation

- Number of nodes
- Frequency of cores
- Size of memories
- Interrupt chaining
- Usage of routing cores or not
- Type of routing core
Hardware Design of a Computation Node C

- Each ARM920T processor runs eCos
eCos - Embedded Configurable Operating System
Hardware Design of a Routing Node $R$

Processors:
- ARM920T
- ARM7TDMI
- ARM926EJ-S
- ARM Cortex A9
- ARM Cortex R4

VIRTUAL BUS

ROUTING NODE

Signal Generator

Program Memory

OS0 ... OS63
Overview

User Input

noc-generator

XML File (Hardware Description)

*.xml

xml-to-sim-model

Hardware Simulation Model

*.h

*.cpp

Executable Simulation Model

*.exe

Run Simulation

Routing Library and Header Files

*.h

*.cpp

Application(s) to run

OVP

Results

Hardware Information Header File

*.h

routing-generator

Run Simulation

OVP

Results

Application(s) to run
xml-to-sim-model

- Generates an OVP simulation model out of the xml hardware description
- Executable model contains interface for (software) applications to run
Open Virtual Platforms (OVP) [Impe 15]
Open Virtual Platforms (OVP) [Impe 15]

- OVP uses a time sliced simulation
Reality

- Core 1
  - Instructions Core 1
- Core 2
  - Instructions Core 2
- Core 3
  - Instructions Core 3
- Peripheral 1
  - Instructions Peripheral 1
- Peripheral 2
  - Instructions Peripheral 2

\[ t_0 = 0 \]
OVP time sliced simulation

- Simulation on one physical core
- \( x \): quantum
- number of instructions per time slice
  \( = \) processor MIPS rate \( \cdot \) time slice duration
- Example:
  \( \text{number of instructions per time slice} = 100\,000\,000 \text{ IPS} \cdot 0.001 \text{ sec} = 100\,000 \)
Overview

The Framework FREACSIM

Measurement and Results

Conclusion

Overview

Application(s) to run noc-generator

User Input

XML File (Hardware Description)

*.xml

Hardware Information Header File

*.h

xml-to-sim-model

Hardware Simulation Model

*.h

*.cpp

Executable Simulation Model

*.cpp

.OVP

Run Simulation

Results

Routing Library and Header Files

*.a

*.h

Application(s) to run

used by

OVP
**routing-generator**

- Creates the applications that run on the routing cores (data exchange)
- Creates a (software) library that can be used by applications that shall run on the computation nodes
- Strategy: Store and Forward Algorithm
- Possible settings for routing
  - Packet size (MTU, Maximum Transmission Unit), 32 to 2048 Byte
How the Routing Works

C0 <-> M0 <-> C1
M1 <-> M2
C2 <-> M3 <-> C3

C: Computation node
M: Shared memory
↔: Bidirectional link (read/write transfers)
How the Routing Works

C : Computation node
R : Routing node
M : Shared memory
: Bidirectional link (read/write transfers)
Overview

The Framework FREACSIM
Measurement and Results
Conclusion
Outlook

User Input

noc-generator

XML File (Hardware Description)

*.xml

xml-to-sim-model

Hardware Simulation Model

*.h

cpp

Executable Simulation Model

*.cpp

Run Simulation

OVP

Results

Application(s)
to run

noc-generator

Interface

User Input

In

Out

Hardware Information Header File

*.h

routing-generator

Routing Library and Header Files

*.a

*.h

used by

Application(s)
to run

OVP

Run Simulation

Results
Outline

1. The Framework FREACSIM
2. Measurement and Results
3. Conclusion
4. Outlook
Applications under Test

- **Bandwidth**
  - sends a MByte of data between the computation nodes and then waits for a single acknowledgement message

- **Stencil Kernel**
  - computes the average of all the direct neighbours of each point in a matrix and writes the result into a second matrix (512×512 of floats)
Applications under Test

- QR-codes
  - Computation core 0 repeatedly rotates the image by some angle and sends the resulting image to the other node cores
  - The other node cores scan the image to detect the QR-Code
  - Image size: 192 KB (256 × 256 pixel RGB)

- Packet Rewriter
  - Computation core 0 creates (artificial) Ethernet packets and sends them to the other computation cores for rewriting in a round-robin fashion
Hardware Used

- **$4 \times 4$ Torus-2D (16 computation and 16 routing nodes)**
Hardware Settings

- Hardware settings
  - Shared memory size: 256 KB
  - Frequency node core: 800 MHz
  - Type of routing core: ARM Cortex-R4
  - Frequency routing core: 500 MHz

- \(800 \text{ MHz} = 800 \text{ MIPS in OVP}\)

- High accurate time slice duration
  - \(\frac{1}{800 \text{ MIPS}} = \frac{1}{800 \text{ IPS}} \cdot 10^{-6} = 0.00125 \text{us} = 1.25 \text{ns}\)
Results Simulation Speed, Packet Size: 64 Byte

Simulated times for different time slices

- time slice 1 us
- time slice 5 us
- time slice 25 us

Use case
- bandwidth
- stencil
- QR
- packet rewriter

Simulated time [sec]
Results Simulation Speed, Packet Size: 64 Byte

Wall clock times for different time slices

- time slice 1 us
- time slice 5 us
- time slice 25 us

Use case
- bandwidth
- stencil
- QR
- packet rewriter

Wall clock time [sec]
Results Simulation Speed, Packet Size: 64 Byte

Average wall clock times per simulated second for different time slices

- time slice 1 us
- time slice 5 us
- time slice 25 us

Use case:
- bandwidth
- stencil
- QR
- packet rewriter
Results Flexibility, Time Slice: 1us

Simulated times for different packet sizes

- 32 Byte
- 64 Byte
- 128 Byte
- 256 Byte

Use case
- bandwidth
- stencil
- QR
- packet rewriter
Outline

1. The Framework FREACSIM
2. Measurement and Results
3. Conclusion
4. Outlook
Conclusion

• Presentation of the highly configurable *Framework for Real-time capable Embedded system and ArChitecture SIMulation (FREACSIM)*

• Fully integrad system-simulation-environment that covers full stack of
  • real-time applications
  • software-based routing
  • NoC/network interconnected specific hardware and architecture aspects for embedded systems

• Speed of instruction accurate simulation

• Flexibility for the user
Outlook

• Validate results concerning to non functional properties (e.g. time)
• Tether a second simulation environment for even more precise results
[Impe 15] Imperas.
“Official Open Virtual Platforms Website”.
Last visit on 01.07.2015.