MPSoC Mapping Exploration by using Calibrated Models

Maximilian Odendahl, Weihua Sheng, Stefan Schürmans, Anastasia Stulova, Jeronimo Castrillon

Map2MPSoC 2011, June 28-29, St. Goar, Germany
Introduction

- MAPS at Map2MPSoC
- Problem

- **MPSoC Mapping Exploration by using Calibrated Models**
  - Abstract Model of PN Applications
  - Calibration of an Abstract Model
  - Automated toolflow

- Results and Evaluation using TI’s OMAP3

- Summary and Conclusion
MAPS: MPSoC Application Programming Studio
- Retargetable, extendable compilation framework
- Light-weight C extension for parallel programming (CPN)
- C-based source-to-source translation for several targets
- Sequential C partitioning
- Scheduling and mapping facilities
- Easy usability through an Eclipse-based IDE

Past presentations at this workshop
- Sequential C code partitioning (2008)
- MVP: High-Level, Virtual Platform (2009)
- CPN: C for Process Networks (2010)
Problem: Software Mapping Exploration is difficult

- **Streaming Applications based on Process Networks**
  - Sequential processes communicate through FIFO channels
  - Splitting computation from communication
  - Extension to C: C for Process Networks (CPN)
  - \textit{cpn-cc} compiler available for various targets using source to source transformation

- **Software Mapping Exploration**

\begin{itemize}
\item GPP
\item DSP
\end{itemize}
Software Mapping Exploration is difficult

- Exponential growth
- Low visibility when targeting embedded systems
- Instruction Set Simulators are too slow

⇒ Replace by Virtual Processing Unit (VPU)

⇒ Abstract models of process network applications needed

⇒ Automated toolflow for creating a fast, accurate, fully functional, virtual prototype by back-annotations
- **Introduction**
  - MAPS at Map2MPSoC
  - Problem

**MPSoC Mapping Exploration by using Calibrated Models**
- Abstract Model of PN Applications
- Calibration of an Abstract Model
- Automated toolflow

- Results and Evaluation using TI’s OMAP3

- Summary and Conclusion
Idea: Annotation and Simulation

- **Use granularity of streaming application**
  - **Computation**: static pattern, use **annotation**
    ```
    rd();
    ...
    for(i=0;i<2;i++){
      ...
      wr();
      ...
      wr();
    }  
    ...
    rd();
    ```
  
  - **Communication**: highly dynamic and dependent on mapping, use **simulation**
    - **Target Binary Trace**
    - **Time Annotated Host Execution**
    - **Calibration DB**
    - **Host Time**
- **Behaviour of PN process** (KPN or subset)
  - Might depend on input data
    - Model has to be functionally correct
  - Only depends on input data
    - Computation parts can be abstracted

- **Abstract model of a PN application**
  - Computation
    - Only local effects
    - Time annotation
  - Communication
    - Might affect whole system
    - Detailed simulation
Calibration of Abstract Model

- **Obtaining Computation Times**
  - Communication: explicit
  - Computation: from end of communication to begin of next one

  ![Diagram showing communication and computation paths]

- **Statistical processing**
  - Store average time for every path

\[
\begin{align*}
\Delta T_A &:= \frac{1}{2} \cdot (\Delta t_1 + \Delta t_4) \\
\Delta T_B &:= \Delta t_2 \\
\Delta T_C &:= \Delta t_3
\end{align*}
\]
Calibration of Abstract Model

- **Abstract Model**
  - Different binary code
  - Basic structure similar to target
  - Communication: explicit

- **Computation**
  - Functional code $\rightarrow$ zero time in simulation
  - Explicit time consumption by annotation
    - Path determined by communication at begin and end
  - Use average time for path

Computation Times

- from $\bullet$ to $\bullet$ $\rightarrow$ Path A $\Delta t_A$
- from $\bullet$ to $\bullet$ $\rightarrow$ Path B $\Delta t_B$
- from $\bullet$ to $\bullet$ $\rightarrow$ Path C $\Delta t_C$
Toolflow

Calibration Phase

- Calibration Mappings
- cpn-cc
- Target Specific Code
- Target Compiler
- HW, simulator or profiler
- Trace Results
- Postprocessing Utility

Streaming Applications (.cpn)

- Exploration Mappings
- cpn-cc
- Abstract Virtual Platform
- Abstract VP Compiler

Calibrated Target: Fast and accurate simulator

- Calibration DB
- Virtual Platform
- Performance, Functional Results

HW, simulator or profiler
Timing on VPU: Computation and Communication

- **Timing**
  - Computation
    - Look up timing in calibration database
  
  - Communication
    - Interconnect of OMAP modeled by a TLM2.0 bus
    - Amount of bus accesses should match
      - Make sure both OMAP and VPU backend use similar FIFO implementations

- **Enhancement of VPU schedulers**
  - New scheduler to track individual thread computation times
MPSoC Mapping Exploration by using Calibrated Models

- **Introduction**
  - MAPS at Map2MPSoC
  - Problem

- **MPSoC Mapping Exploration by using Calibrated Models**
  - Abstract Model of PN Applications
  - Calibration of an Abstract Model
  - Automated toolflow

- Results and Evaluation using TI’s OMAP3

- **Summary and Conclusion**
Evaluation using TI OMAP

- **Show feasibility of approach**
  - Run PN application on real HW platform
  - Create abstract model + calibrate it
  - Compare estimated times to real times

- **HW platform**
  - TI OMAP

- **Abstract VP**
  - 3x VPU
  - TLM 2.0 bus
Results: Benchmarks and Error Sources

- **Benchmarks:**
  - Scharr
  - Mandelbrot
  - MJPEG

- **Error Sources:**
  - Computation time
  - Communication time
  - Scheduler model

\[
error = \left| \frac{time_{VPU} - time_{OMAP}}{time_{OMAP}} \right|
\]
Results

Scharr Application

Mandelbrot Application

MJPEG Application

0 5 10 15 20 25 30 35 40
Relative Error in %

0 5 10 15 20 25 30 35 40
Relative Error in %

Mapped to ARM A8
Mapped to C64x
Introduction
- MAPS at Map2MPSoC
- Problem

MPSoC Mapping Exploration by using Calibrated Models
- Abstract Model of PN Applications
- Calibration of an Abstract Model
- Automated toolflow

Results and Evaluation using TI’s OMAP3

Summary and Conclusion
Summary

- **Automated toolflow to add an accurate SW performance model to a high-level, virtual platform**
  - Does not give you a perfect mapping configuration
  - But: all advantages of a calibrated, virtual platform
  - Making software mapping exploration a lot easier

- **Automated calibration**
  - Only a high-level model of the real platform is needed
  - Creation of new hardware models with reasonably low effort
Summary and Conclusion

- **Need for abstract models of PN applications**
  - PN applications are run on MPSoCs
  - Abstract VPs are used for MPSoCs exploration

- **Automatic generation of abstract models**
  - Functionality taken from high-level PN code
  - Communication modeled in detail
  - Computation modeled by time annotation

- **Calibration of abstract models**
  - Measurements of application on HW
  - Communication as synchronization points

- **Approach is feasible**
  - Implementation needs some improvement
Thank you for your attention!

Questions?