Hardware Specific Code Generation at Different Abstraction Levels using Averest

Omair Rafique, Manuel Gesell and Klaus Schneider

gesell,schneider@cs.uni-kl.de
es.cs.uni-kl.de

Embedded Systems Group
University of Kaiserslautern

16th International Workshop on Software and Compilers for Embedded Systems

21.06.2013
Contents overview

- Introduction
- Case Study
- Code Generation Using Averest
- Summary
Contents

- Introduction
  - Abstraction Levels
  - The Averest Toolkit

- Case Study

- Code Generation Using Averest

- Summary
Modeling an Embedded System

- **Software Design Levels**
  - resource management
  - four abstraction levels
    - pure hardware development
      - application specific hardware circuit
      - resources need to be developed
    - bare-iron design
      - embedded processor
      - resource management without OS
    - OS-based design
      - embedded processor
      - resource management using OS
    - model-based design
      - realization-independent models
      - no consideration of resources
Abstraction Levels (1)

- Pure Hardware Development
  - application specific hardware circuit
  - resource management
    - needs to be developed
  - Advantages:
    - small, efficient and fast component
  - Disadvantages:
    - long development period
    - inflexible results
    - fixing errors
    - maintenance
Abstraction Levels (2)

- **Bare-Iron Design**
  - represents system as a software
  - no operating system
  - resource management
    - memory mapped input/outputs
- **Advantages:**
  - reduces manufacturing cost
  - error fixing is cheaper
- **Disadvantages:**
  - insensitive production cost
  - may require software re-writing
Abstraction Levels (3)

- **OS-Based Design**
  - requires an operating system
  - resource management
    - multitasking, scheduling etc.
    - input/output drivers
  - **Advantages:**
    - better conceptualization
    - better management
  - **Disadvantages:**
    - scheduling issues
    - race conditions
    - mutual exclusion
Abstraction Levels (4)

- **Model-Based Design**
  - certain level of detail
  - targets exploitation of the abstract representation
  - realization-independent behavior
  - resource management
    - no consideration for resource management
  - introduction of hardware abstraction layer (HAL)

- **Advantages:**
  - new revision manageable
  - high computational requirements

- **Disadvantages:**
  - moving logic requires effort
Introducing Averest

http://www.averest.org
Contents

- Introduction

- Case Study
  - The Purpose
  - The Architecture
  - The Driving Mechanism
  - Modeling ECUs Behavior

- Code Generation Using Averest

- Summary
The ConceptCar, Purpose
The ConceptCar, Architectural View

What is Concept Car?
- embedded system
- radio controlled transmission system
- same structure as a modern car
- different processing units
- two separate power sources
- separate EmergencyBoard
The ConceptCar, Driving Mechanism

1. Radio Remote Receiver
2. CAN Bus
3. ARMBoard
4. SensorBoard Inertial ECU
5. DistanceBoard ECU
   - ActorBoard ECU
   - SensorBoard Throttle ECU
   - SensorBoard Steering ECU
   - EmergencyBoard ECU
   - Servo and DC Motor

Throttle
Steering
The ConceptCar, Modeling Electronic Control Units (ECUs)

- ECU models at different abstraction levels
  - SensorBoard Throttle and Steering
    - bare-iron level
  - SensorBoard Inertial
    - model-based design
  - ActorBoard
    - bare-iron level
    - combination of model-based design and RTOS
The ConceptCar, at a glance

http://conceptcar.iese.de
Contents

- Introduction
- Case Study
- Code Generation Using Averest
  - The Work Flow
  - Targeting Different Abstraction levels
- Summary
The Code Generation Work Flow

Quartz Code → Averest Toolkit → C Code (Model-Based Behavior) → Hardware Abstraction Layer (HAL)

Quartz Code → Averest Toolkit → C Code (OS-based Behavior) → Hardware Abstraction Layer (OS)

Quartz Code → Averest Toolkit → C Code (bare-metal Level Behavior) → Hardware Abstraction Layer (Baremetal)

Verilog Code (Pure Hardware Development) → State of the Art Tool → C Compiler → Executable

Technical University of Kaiserslautern 6/21/2013
Targeting Abstraction Levels (1)

- Quartz description to hardware description
  - compilation (AIF generation)
  - AIF transformation (construction of an equation system)
  - HW synthesis (translation to Verilog code)
- state of the art tools
Targeting Abstraction Levels (2)

- Quartz description to software description
  - compilation (AIF generation)
  - implicit transformations
  - explicit transformation
    - ToBareIron
    - ToOS
    - ToModel
  - SW synthesis (translation to C code)

- (Abstraction Level Specific) Hardware Abstraction Layer (HAL)
Contents

- Introduction
- Case Study
- Code Generation Using Averest
- Summary
Summary

- Conclusion
  - an approach to target different levels of abstraction
  - Averest work flow
  - ConceptCar as a case study
  - Hardware Abstraction Layer (HAL)
  - the outcome
THANK YOU FOR YOUR ATTENTION!!

Any Comments,
Feedback,
Questions?