EECS 298: System-on-Chip Description and Modeling Lecture 8

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 8: Overview

- Homework Assignment 2
  - Discussion
- Homework Assignment 3
  - Tasks
  - Discussion
- System Validation using SpecC
  - Simulation
  - Debugging
  - Tracing

Homework Assignment 2: Discussion

- Project
  - Elevator Control System (ECS)
    - Distributed embedded system
    - Set of communicating Elevator Control Units (ECU)
- Tasks for System Specification
  - Decompose ECS into multiple ECUs
  - Develop a specification model for each ECU
  - Validate each ECU model using simulation
  - Compose entire ECS using developed ECUs
  - Validate entire ECS
  - Then, refine and implement ECS...

Homework Assignment 2: Discussion

- Decomposition of ECS
  - Floor panel
    - panel at each floor and each shaft with up/down controls
  - Floor display
    - display of current floor and direction at each floor
  - Floor door
    - Control unit to open/close doors at each floor
  - Car panel
    - panel in each car with request controls
  - Car display
    - display of current floor and direction in each car
  - Car door
    - Control unit to open/close doors in each car
  - Main control unit
    - central control unit to control the entire ECS
  - Motor control unit
    - control unit for the motor atop each shaft

Homework Assignment 2: Discussion

- Deliverables
  - Specification document for one ECU
    - Illustration figure (e.g. FloorDoor.pdf)
    - Schematic view of ECU SoC with ports
    - Brief (!) description of functionality (in English)
  - Executable specification model for one ECU embedded in proper test bench (e.g. FloorDoor.sc)
    - SpecC source code
  - Successful simulation run (e.g. FloorDoor.log)
- Due
  - ~~Week 5 (next week)~~ March 1, 2006, 11:59pm
    Email to **doemer@uci.edu**

Homework Assignment 3

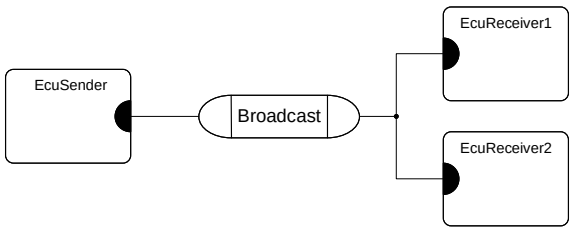
- Task
  - Communication refinement of chosen ECU
    - Transaction Level Model (TLM)
    - Bus Functional Model (BFM)
- Components
  - Controller Area Network (CAN) Bus
    - provided SpecC models
      - TLM (file **canEcu_tlm.sc**)
      - BFM (file **canEcu_bfm.sc**)
    - see file **/home/eecs298w06/canEcu.tar.gz** on server **epsilon.eecs.uci.edu**

Homework Assignment 3

- Controller Area Network (CAN) Bus
  - Properties
    - Standard bus used in automotive industry (Bosch GmbH)
    - Serial, multi-master, broadcast communication protocol
    - Collision-avoidance arbitration (fixed priorities)
    - Built-in synchronization and error detection
    - Single wire protocol (pull-down mechanism)
  - Modeling in SpecC
    - G. Schirner, R. Dömer:
      "Abstract Communication Modeling: A Case Study Using the CAN Automotive Bus",
      IESS, August 2005.

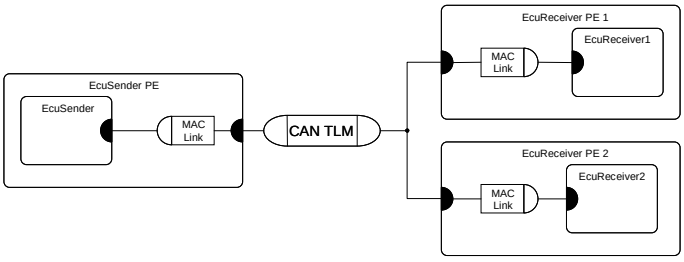
Homework Assignment 3

- Controller Area Network (CAN) Bus
  - Example
    - Abstract communication model
      (specification model)



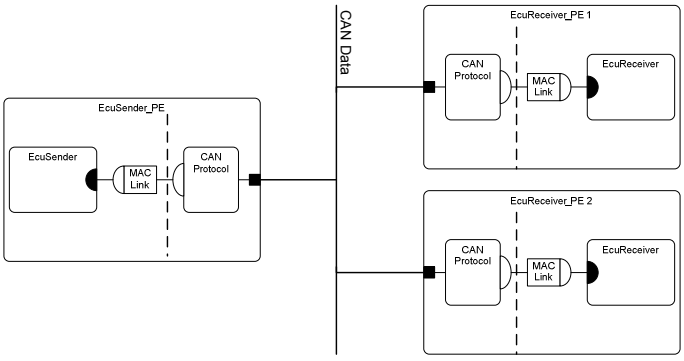
Homework Assignment 3

- Controller Area Network (CAN) Bus
  - Example
    - Transaction level model (file **`canEcu_tlm.sc`**)



Homework Assignment 3

- Controller Area Network (CAN) Bus
  - Example
    - Bus-functional model (file **`canEcu_bfm.sc`**)



Homework Assignment 3

- Deliverables
  - Documentation
    - Schematic view of refined ECU models
    - Brief (!) description of functionality (in English)
    - e.g. **`FloorDoor_TLM.pdf`**, **`FloorDoor_BFM.pdf`**
  - Refined ECU models in proper test bench
    - SpecC source code
    - e.g. **`FloorDoor_TLM.sc`**, **`FloorDoor_BFM.sc`**
  - Successful simulation run
    - e.g. **`FloorDoor_TLM.log`**, **`FloorDoor_BFM.log`**
- Due
  - March 8, 2006, 11:59pm
  - Email to **`doemer@uci.edu`**

System Validation using SpecC

- Simulation
  - **`scc DesignName –sc2out –vv –ww`**
    `./DesignName`
  - Header file **`sim.sh`**
    - Access to simulation time
      - macros **`PICO_SEC`**, **`NANO_SEC`**, **`MICRO_SEC`**, **`MILLI_SEC`**, **`SEC`**
      - typedef **`sim_time`**, **`sim_delta`**, **`sim_time_string`**
      - function **`now()`**, **`delta()`**
      - conversion functions **`time2str()`**, **`str2time()`**
    - Handling of bit vectors
      - conversion functions **`bit2str()`**, **`ubit2str()`**, **`str2bit()`**, **`str2ubit()`**
    - Handling of long-long values
      - conversion functions **`ll2str()`**, **`ull2str()`**, **`str2ll()`**, **`str2ull()`**

System Validation using SpecC

- Debugging
  - **`scc DesignName –sc2out –vv –ww –g -G`**
    `gdb ./DesignName`
    `ddd ./DesignName`
  - Header file **`sim.sh`**
    - Access to simulation engine state
      - functions **`ready_queue()`**, **`running_queue()`**, etc.
      - functions **`_print_ready_queue()`**, **`_print_running_queue()`**, etc.
      - function **`_print_process_states()`**
      - function **`_print_simulator_state()`**
    - Access to current instance
      - functions **`active_class()`**, **`active_instance()`**
      - functions **`current_class()`**, **`current_instance()`**
      - functions **`print_active_path()`**, **`print_current_path()`**
      - ...

System Validation using SpecC

- Tracing
  - **`scc DesignName –sc2out –vv –ww -Tvcds`**
    `./DesignName`
    `gtkwave DesignName.vcd`
  - Trace instructions in file **`DesignName.do`**
  - Trace log in file **`DesignName.vcd`**
  - Waveform display **`gtkwave`**
    - available as **`/opt/gtkwave/bin/gtkwave`**