Chip simulation used to run automotive software on PC

Jakob Mauss, QTronic GmbH

ERTS 2014
Embedded Real Time Software and Systems
Toulouse, France
February 5 - 7, 2014
Outline of the talk

Chip simulation of automotive ECUs

1. Motivation
2. Chip Simulation
3. Applications
4. Limitations
5. Conclusion
Motivation

typical development project: **100 person years**
Motivation

typical development project: **100 person years**

Chip simulation used to run automotive software on PC - ERTS 2014, Toulouse
Motivation

Typical development project: **100 person years**

- need to parallelize development: HOW?
  - move tasks to a virtual development environment
  - duplicate for each developer: easy, cheap, no blocking
  - protect IP
How to simulate an ECU on PC?

Three Options

1. **retarget** the C Code of the ECU: compile for Windows
   - requires access to C code: typically owned by supplier, not OEM

2. **reverse engineer** selected ECU functions, e.g. with Simulink
   - time consuming, error prone, possibly illegal

3. **rehost** the binary code (hex file) to Windows PC:
   - aka chip simulation, to be presented here
Example chip: TriCore TC1797

processing of about 850 instructions

Source: Infineon AG

On-Chip Devices
ADC Analog Digital Converter
DMA Direct Memory Access
EBU External Bus Unit
FADC Fast Analog Digital Converter
GPTA General Purpose Timer Array
MultiCAN CAN Controller
MSC Micro Second Channel
PCP2 Peripheral Control Processor v2
Ports General Purpose I/O Ports and Peripheral I/O Lines
SCU System Control Unit
SSC Synchronous Serial Interface
STM System Timer

Chip simulation used to run automotive software on PC - ERTS 2014, Toulouse
Requirements: Chip sim. for Automotive

Virtual System Prototypes (VSPs)
- chip-specific, rich model:
  all on-chip devices modeled
- no support for automotive interfaces
- expensive: > 100.000 Euro

Required by
automotive SW developers and
calibration engineers
- support for automotive interfaces: A2L, MDF/DAT, XCP, DBC, ...
  - to reuse existing files
  - to interface with existing tool chain
- inexpensive: < 10.000 Euro
- easy to setup: no (or very little) chip-specific knowledge available
A "virtual ECU" for PC

**Silver** (a QTronic product) turns a hex file into a "virtual ECU"

Either a
- Simulink SFunction
- FMU for Co-Simulation
- Silver module (DLL)

**Principles:**
- All instructions emulated: bit-accurate results
- Nearly no on-chip devices emulated: use inputs and outputs to by-pass basic SW
- Run only the functions of interest: fast!

**TriCore emulation:** 60 MIPS
**PowerPC emulation:** 200 MIPS

**on-line calibration:** measure and tune running simulation

**virtual ECU**
- ECU functions
- RTOS emulation
- 4GB virtual memory
- A2L conversion
- XCP

**TCP/IP**

INCA or CANape

vehicle simulation or measurements

Silver or Simulink or any FMU Simulator

Chip simulation used to run automotive software on PC - ERTS 2014, Toulouse
Example 1: virtual ECU in Simulink

all ECU characteristics are automatically turned into MATLAB workspace parameters
default values taken from hex file
passed to the Sfunction as parameter
Example 2: virtual TCU running in Silver
Setting up a Chip simulation

1. write spec.txt to specify what functions to run
2. step and debug the simulation in Silver debug mode
3. generate fast running SFunction, FMU or Silver module
Setting up a TriCore simulation

1. write spec.txt to specify what functions to run
2. step and debug the simulation in Silver debug mode
3. generate fast running SFunction, FMU or Silver module

```plaintext
# specification of virtual ECU
hex_file(m12345.hex, PowerPC)
a2l_file(m12345.a2l)
map_file(m12345.map)  # a TASKING or GNU map file
chip_config(STEP_SIZE, 10)  # base clock tick in ms
chip_config(USER_STACK, 0xd0001000)  # location of user stack

# functions to be simulated, in order of execution
task_initial(ABCDE_ini)
task_initial(ABCDE_inisyn)
task_triggered(ABCDE_syn, trigger_ABCDE_syn)
task_periodic(ABCDE_20ms, 20, 0)
task_periodic(ABCDE_200ms, 200, 0)

# interface of the generated Sfunction, FMU, or Silver module
input(nmot)
output(target_cur_mv1)
output(target_cur_mv2)
parameter(mv_curve)
```
real RTOS: at t=23ms, a **10 ms task** is interrupted by an **event triggered task**. At that time, the context of the interrupted task needs to be saved. Execution of a task takes time.

Silver: Execution of a task takes **zero time**. Tasks run at scheduled execution time and execute 'infinitely fast': task execution can therefore not be interrupted: No need for context switch.
Applications and Limitations

Applications reported by Daimler and IAV

- mathematical optimization of engine maps
- on-line calibration with INCA and CANape
- PC test of manipulated hex files

Limitations

- instruction accurate, but not cycle accurate: cannot predict exact execution times
- based on TriCore and PowerPC specification: 'silicon bugs' are not simulated
- on chip devices not modeled: cannot run basic software, such as device driver
Summary

Virtual ECUs for Windows PC
- no expensive reverse engineering
- no C code required
- based on available HEX, MAP and A2L file
- excellent integration with automotive tool chain
- high accuracy of model
- application example: shift calibration tasks to PC

- works currently for
  TriCore processors: TC1796, TC1797, TC1798, ... Aurix: 60 MIPS
  PowerPC MPC55x and Qorivva chips with e200 cores: 200 MIPS

Meet us at the ERTS EXPO: Stand #50