CS4272: Hardware Software Codesign

Introduction

Abhik Roychoudhury
School of Computing
National University of Singapore

The Course Title

- Much of this lecture will explain the course title.
- Roughly speaking
  - An advanced course on Embedded System (ES) design
    - An advanced course on ES design from design models to software and hardware.
      - An advanced course on ES design from design models to software and hardware satisfying constraints on time, power, code size

Today's Lecture

- Embedded Systems (ES)
- Quick tour/re-cap of current issues in ES design.
- Overview of co-design methodologies
- Course structure, assessment etc.

ES

- A computing system which is part of a "larger system" (read – device).
- The larger system constitutes the environment – in continuous interaction.
- The computing system implements a specific functionality.
  - A dedicated computer implemented by a combination of hardware and software.

ES Examples

- Automobiles
- Train control systems
- Avionics / Flight control
- Nuclear Power Plants
- Inside medical devices (for image manipulation) and other purposes
- Safety first !

ES Examples

- Or more vanilla
- HDTV
- Washing Machines
- Microwave
- Controllers for other household devices such as Air-con
- Finally, smart room / wear (GA Tech etc.)

Examples

Some embedded systems from real life

Pedometer

- Obvious computer work:
  - Count steps
  - Keep time
  - Averages
  - etc.
- Hard computer work:
  - Actually identify when a step is taken
  - Sensor feels motion of device, not of user feet

Mobile phones

- Multiprocessor
  - 8-bit/32-bit for UI
  - DSP for signals
  - 32-bit in IR port
  - 32-bit in Bluetooth
- 8-100 MB of memory
- All custom chips
- Power consumption & battery life depends on software

Mobile base station

- Massive signal processing
  - Several processing tasks per connected mobile phone
- Based on DSPs
  - Standard or custom
  - 100s of processors

Telecom Switch

- Rack-based
  - Control cards
  - IO cards
  - DSP cards
  - ...
- Optical & copper connections
- Digital & analog signals

Smart Welding Machine

- Electronics control voltage & speed of wire feed
- Adjusts to operator
  - kHz sample rate
  - 1000s of decisions/second
- Perfect weld even for quite clumsy operators
- Easier-to-use product, but no obvious computer

Sewing Machine

- User interface
  - Embroidery patterns
  - Touch-screen control
- "Smart"
  - Sets pressure of foot depending on task
  - Raise foot when stopped
- New functions added by upgrading the software

Forestry Machines

Forestry Machines

- Networked computer system
  - Controlling arms & tools
  - Navigating the forest
  - Recording the trees harvested
  - Crucial to efficient work
- Processors
  - 16-bit processors in a network

Operator Panel

- Embedded PC
  - Graphical display
  - Touch panel
  - Joystick
  - Buttons
  - Keyboard
- But tough enough to be "out in the woods"

Cars

- Multiple processors
  - Up to 100
  - Networked together
- Multiple networks
  - Body, engine, telematics, media, safety

Cars

- Functions by embedded processing:
  - ABS: Anti-lock braking systems
  - Airbags
  - Efficient automatic gearboxes
  - Theft prevention with smart keys
  - Blind-angle alert systems
  - ... etc ...

Cars

- Large diversity in processor types:
  - 8-bit – door locks, lights, etc.
  - 16-bit – most functions
  - 32-bit – engine control, airbags
- Also, note that
  - Processing where the action is
  - Sensors and actuators distributed all over the vehicle

A note about cars

- Car electronics is an increasingly important market, requiring new design flows.
  - Software is important for value addition
- Comments by major manufacturers
  - Daimler Chrysler
    - More than 90% of the innovation is from the car electronics (and not from the mechanical parts!)
  - BMW
    - More than 30% of the manufacturing cost of a car is from the electronic components !
- Reliable/robust ES design flows needed !

Car electronics

- 1. Critical features in the power train or chassis
  - Control engine, brakes, steering wheel
  - Safety-critical, hard real-time
  - Accomplished by communicating Electronic Control Units (ECUs) which contain
    - Micro-controller(s), RTOS, application program
    - ECUs communicate via buses
    - Communication between diff. micro-controllers in the same ECU also supported by dual-ported RAMs
    - Protocol design issues for the bus communication
    - … all in the CS/CE domain, as you can see

Car Electronics

- 2. Controlling Cabin features
  - Power windows, air-conditioning
  - Often given as complex state-based specifications which get translated to code
- 3. Infotainment/Telematics
  - Relates to Entertainment, not critical
  - Soft real-time constraints
  - Protocol standards for communication among media devices in a network …

Finally, if you want to play

- Lego mindstorms robotics kit
  - Standard controller
    - 8-bit processor
    - 64 kB of memory
  - Electronics to interface to motors and sensors
- RCX Programmable controller, see
  - http://legolab.daimi.au.dk/CSaEA/RCX/Manual.dir/RCXManual.html

Today's Lecture

- Embedded Systems (ES)
- Quick tour/re-cap of current issues in ES design.
- Overview of co-design methodologies
- Course structure, assessment etc.

ES Characteristics

- Real-time and/or Reactive
  - Often combines hard and soft real-time
  - Timing constraints on the response
- Low power budget
  - Novel architectures etc.
- Low code size
  - Aggressive Code compression possible.
- Profile driven development all important.

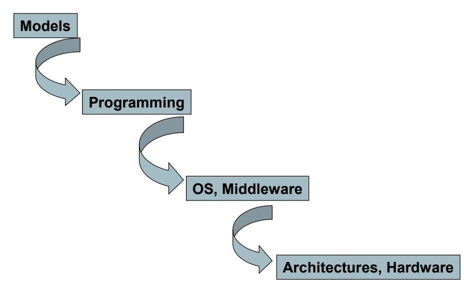
All methodologies …

- Will typically espouse
  - Integrated model based design
  - Efficient Programming Technology
    - Design Patterns, Run-time support
  - Link up to middleware/ RTOS
    - Services to Application Software
  - System-on-Chip Platforms and/or Programmable hardware
  - High levels of assurance at each stage.

But this is …

- A little bit too abstract
- Sounds like canonical CS stuff or some collections of it.
  - What are the ES design issues ?
- We need to look deeper and in more details.

Levels of Abstraction



Models of Computation

"The primary technology outcome of the MoBIES program will be Model-Based Integration technology.

The key technology components to be developed are model-based programming environments and model-based generators. … "

Excerpt from US Air Force Research Lab
MOBIES Program Objectives

Models of Computation

- Model based programming
  - Programming Systems which allow construction of complex models as a by-product of programming
    - not the current norm !
- Model based generators
  - Input to verification/analysis tools from models
  - Generate actual component interfaces etc. from models

What kind of models ?

- Concurrent processes
- Support for timing
- Support for process communication
  - Explicit modeling of process communication can ease interface synthesis.
  - Separation of computation /communication
- Objects (crucial link to Programming Technology)
- Discrete and Continuous models

Programming Technology

- Hard and soft real-time requirements
- Inter-process synchronization requirements
  - Implementation of communication
- Communication is often peer-to-peer rather than centralized.
- All these requirements need to introduced into otherwise vanilla code.

Programming Technology

The PCES program will extend and combine approaches from programming language analysis and compilation; composable policies and protocols for communications and operating systems services, and correct-by-construction software techniques. These techniques will be developed for both interactive and automatic use in support of embedded real-time programming and execution.

Excerpt from the "Vision" of DARPA PCES Program

- Program Composition for Embedded Systems

Or more simply …

- Develop re-usable core software.
- Develop list of "features" (or "aspects") you want to assure
  - Synchronization , Timing, Memory management, Power
- Safe code transformation for introducing these aspects into core code.
- Analysis tools for studying tradeoffs between cross-cutting aspects.
- AOP, SW Composition: much to learn from PL,SE

OS, Middleware

"The DARPA Quorum program is pursuing technology research projects that are attempting solutions to a number of the missing capabilities needed for mission critical system development, such as predictable performance for network based applications, fault tolerance and dependability characteristics, real time performance properties, and fine grained distributed systems security. "

Excerpt from description of DARPA Quorum Program

So, the key points include…

- Guarantee of real-time performance (related to scheduling and schedulability)
  - Related to timing and/or power estimation
- Quality of Service (QoS) guarantees in networked distributed embedded systems
  - Handling multiple QoS guarantees – expt. Platform
  - Formal specification of QoS guarantees and even run-time monitoring.
- Middleware for open systems
  - Fault tolerance, Dynamic scheduling.

Hardware, Architecture

- Low power processor architectures
- Power model for system-on-chip platforms
  - Need to model buses / networks, and traffic flow
  - Abstractions in traffic flow : count based distributions
- Transforming assembly code to reduce power hotspots (more mem. Accesses etc.)
- Other layers can also handle power
  - E.g. power as a secondary concern in scheduling
  - But here power is all important.

Hardware, Architecture

- Transformations for predictable code
- Take into account underlying hardware platform (a formal model of pipeline, cache etc. ?)
- Both sides of the coin
  - Analyze the performance of assembly code to find worst- case execution time etc. (Well understood)
  - Trade off performance for predictability
    - Cache locking schemes, New memory architectures
  - Develop models to predict the effect of well-understood code transformations

Hardware, Architectures

- Other conventional areas in co-design
  - Other EDA topics (like design space exploration) conventionally considered under HW/Arch., …
  - … but can be considered to belong to all the layers of ES design presented in this talk.
- Reconfigurable architectures
  - Reconfigure data-paths, functional units etc. at run-time and/or based on application.
  - Technical support – Field Programmable Gate Arrays (FPGA): plug-n-play with processors in a design.

Organization

- Embedded Systems (ES)
- Quick tour/re-cap of current issues in ES design
- Overview of co-design methodologies
- Course structure, assessment etc.

Co-design (1)

- A. Model the system as a whole
  - Irrespective of which parts are implemented in hardware and which parts in software.
  - Various choices of Models of Computation for reactive real-time systems.
    - Model level functionality verification is possible.
- B. Partition into HW and SW
  - HW: Can be reconfigurable (FPGA)
  - SW: Run on micro-controllers or more complex processors.
    - Further allocation needed if multiple processing elements (PEs) are available.

Co-design (2)

- C. Scheduling
  - After allocation of tasks to PEs
  - Determines order in which tasks allocated to the same PE will be invoked so that
    - Performance constraints (deadlines) are met.
    - Any dependencies between tasks are preserved.
    - Communication/context-switch overheads in execution are minimized if possible.
  - Requires execution-time estimation of the tasks as well.
    - However, this involves the code which may require further optimization, not purely linear design flow !

Co-design (3)

- D. Communication synthesis
  - Simple: Replace shared var. names by appropriate locations
    - If placed in local memory of a PE, the other PEs need to access this.
    - Choice of memory hierarchies for each PE --- there might be a unified or a per-process scratchpad memory where shared variables are allocated leading to different communication overheads.
  - Complex: Design interfaces to enable communication among design components
    - Native protocols of the design components are incompatible, but can be "fixed" !

Co-design (4)

- E. Implementation
  - E1. SW implementation
    - Compilation of the SW parts of the design
    - Many issues in optimization of embedded software to reduce code-size, energy consumption as well.
    - At this stage, also profile and debug the code.
  - E2. HW implementation
    - Convert behavioral design components to netlists at Register Transfer level.
    - Can be run on ASIC or FPGA.
  - E3: Interface implementation
    - Convert interfaces designed in the previous step to (typically) hardware ASIC implementation.

Co-design (5)

- Iterative process
  - If performance/power constraints are not met, we need to modify design choices
    - Design space exploration
  - To reduce # of iterations, try to make smart choices straightaway
    - e.g. during partitioning try to get optimal partitioning
    - The design space is very large
      - **Architecture --- choice of PEs, memory in PEs, allocation of tasks to PEs**
      - Partitioning of a problem into software/hardware
      - Even, choice of optimizations for a given code
    - Different search strategies to traverse (restricted parts) of this design space !

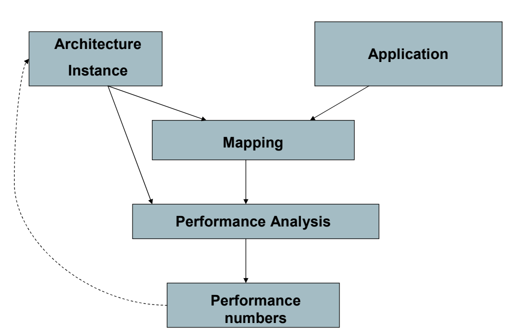
Choosing the Architecture

- Fix a specific architecture
  - Processors/memory for each PE etc
  - When we described the co-design flow, we implicitly assumed as if the PEs are all fixed and available --- this is not the case usually.
- Let the architecture emerge from the appl
  - As the application is modeled and then gradually converted to impl., the arch. is fixed
- Fix a family of architectures
  - Design space exploration and design point estimation within this family.

What is an Arch. Family?

- Family of similar architectures
  - Parameter values define instances within the family.
- Example
  - An in-order pipelined processor with direct-mapped cache.
  - Instance defined by
    - # of pipeline stages
    - # of cache lines
    - Cache line size

Y-chart approach



Organization

- Embedded Systems (ES)
- Quick tour/re-cap of current issues in ES design
- Overview of co-design methodologies
- Course structure, assessment etc.

Learning Objectives

- Embedded system design from high-level executable specifications
  - Combined model for mixed (HW + SW) solutions.
- Techniques for selecting implementation platform and mapping an application to a given platform
- Formal analysis of key criteria like performance.
- Pre-requisites
  - CS2271 and CS3212

Readings

- Lecture notes & Weekly readings
  - Check the course web-page
    - http://www.comp.nus.edu.sg/~cs4272
- Text Book
  - Embedded System Design by Peter Marwedel, Springer.
  - Available in Co-op
  - Covers the topics mostly --- supplemented via lecture notes.

Assessment & Contact

- Final : 45%
- Midterm: 25%
  - To be held in the 7th week
- Programming Assignments (3 in total)
  - 30%
- Contact details & consultation
  - My office is in COM1 #03-20.
  - My e-mail: abhik@comp.nus.edu.sg
  - Follow details from course web-page
    - http://www.comp.nus.edu.sg/~cs4272