Design, Construction and Verification of Embedded Real-Time Systems:

Some Research Challenges

Raj Rajkumar
ECE and CS
Real-Time and Multimedia Systems Laboratory
Carnegie Mellon University
raj@ece.cmu.edu
http://www.ece.cmu.edu/~raj

Outline

- Motivation
- Research Challenges
- Missing Technology Pieces
- Concluding Remarks

Motivation

- Large numbers of automobiles
  - Drive-by-wire, brake-by-wire, X-by-wire…
- Large volumes of devices
  - cell-phones, home/building monitors, sensor networks
- Embedded everywhere…
  - Pervasive/ubiquitous computing
- Large-scale real-time platforms
  - Assembly/manufacturing plants, military ships, avionics, …
- ...

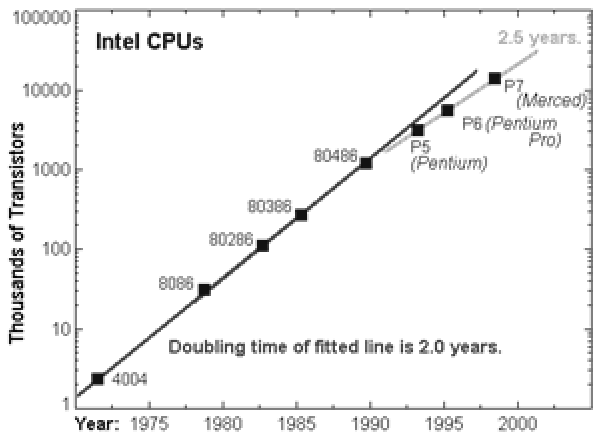
The Goal

- How do we ensure that these systems can be
  - Built correctly?
    - Rather, *mostly* correctly
      - Get the important pieces correct
  - Deployed cost-effectively?
  - Maintained easily?
- Add in the unexpected
  - S*** happens
  - Worm-mongers, virus developers, …
  - Really bad guys want to mess things up real badly…

An Historical Perspective

Chip Technology Growth

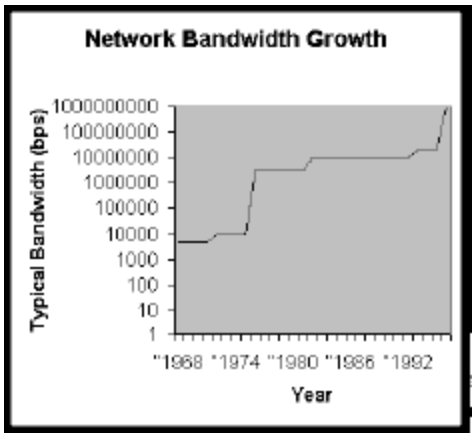
- Moore's Law (of course)



Storage Technology Growth

- Disk density has nearly doubled every 15 months for the past 5-6 years
  - Prices have dropped more sharply than processing power

Network Technology Growth



Productivity of Embedded Software Development

- Very few hardware designers design using transistors, or gates
  - Why do software programmers?
- Can hardware programmers get better run-time efficiency by designing it all from scratch?
  - Yes, but they would have to be crazy
- The modern assembly-line:
  - Software programmers editing ASCII files (in a colorful IDE)

→We software designers and developers are crazy!

→Programming in ASCII-based programming languages considered harmful

Core Research Challenges

Core Research Challenges

- Bringing '-ilities' together
  - Comprehensive modeling frameworks
  - QoS and multi-resource management
    - Energy management
  - Practical verification techniques
  - Automation

Need: "The Unified Theory of Embedded Systems"

- Real-time and resource management
  - Rate-monotonic analysis (RMA), EDF, …
  - QoS-based resource allocation (Q-RAM, …
- Fault-tolerance and high availability
  - Replica management (active/TMR, primary/backup, …)
  - Fault models (Fail safe, Byzantine,)
- Security
  - Authentication, authorization, encryption, …
- Size and Energy Constraints

How do we bring all of them together?
Ideally, properties are composable…
And only modify parameters along other dimensions

Example: RT & FT Q&A (1)

- What is the state-of-the-art in RT-FT distributed systems?
  - Deployed RT-FT systems are "**customized**" – build only for the application at hand
- What can we *not* do today in building RT-FT systems?
  - Be flexible, reusable, low-cost, composable, ...
  - Seamless operation as more and more resources fail (or join)
- What are some urgent issues to address in order that we can build such systems?
  - Need to **re-visit most FT abstractions**
    - Kinds of data consistency
    - Kinds of replication
    - Implications to timing properties

RT & FT Q&A (2)

- What is it going to take for us to solve these problems?
  - Conceptual framework with **theoretical underpinnings**
  - **Challenge problems** to drive focused research
- What are the obstacles to achieving these goals?
  - Two "distinct and separate" communities
  - Specialized needs of the past
  - "Niche" (and therefore "small") markets

RT & FT Q&A (3)

- What is a long-term vision (i.e., where is research headed, and what kinds of problems will we solve)?
  - The problem will become mainstream...
  - It's a system-level problem...
  - ... and the problem will be solved!
  - Predictability and analyzability
  - Composability, re-usability and testability
  - Hardware and software interfaces
  - Application frameworks (e.g. adaptability)

Core Research Challenges

- Bringing '-ilities' together
- Comprehensive modeling frameworks
- QoS and multi-resource management
  - Energy management
- Practical verification techniques
- Automation

The Power of Abstractions

- Diodes
- Transistors
- Gates
- Chips
- Boards
- “Boxes”
- Systems

The driving force is the power of the abstraction and its impact on productivity and ease of usage.

Not performance (thanks to Moore’s Law)

Programming Abstractions

- Machine languages
- Assembly languages
- High-level languages
- Object-oriented languages
- Components

What’s next?

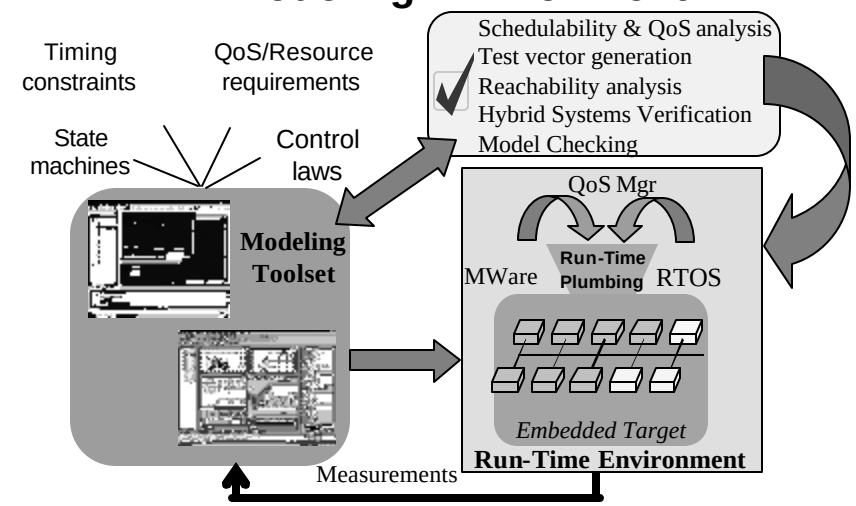
- Modeling languages and environments

Software Development Hourglass

- A design-time hourglass
  - Not just for run-time technologies

E-commerce Manufacturing Embedded Defense
Architectural rules Patterns Subsystems
Models
Java C++ C CORBA VB
Windows Linux VxWorks uCOS
Pentium PowerPC MIPS ARM SH XScale

Modeling Environment

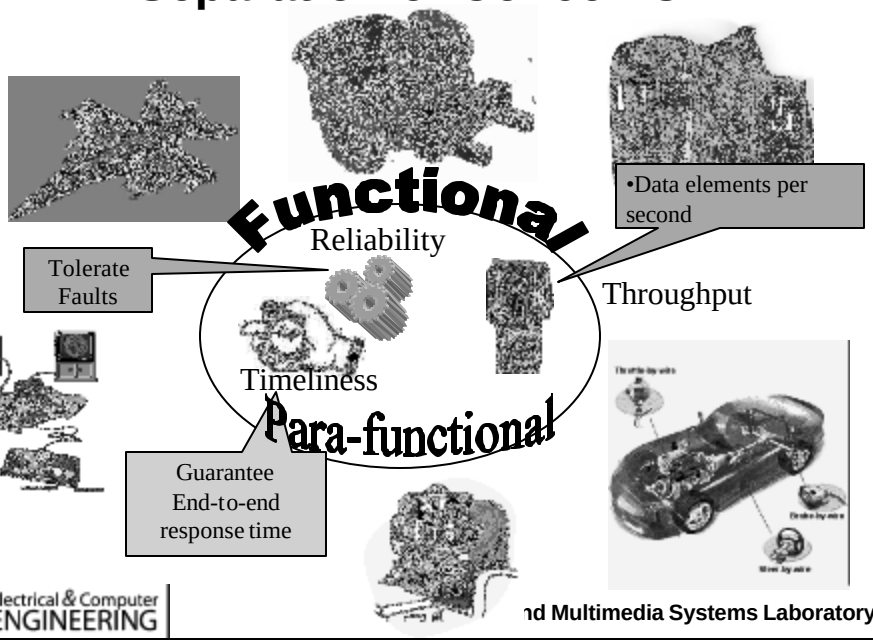


Simultaneous and consistent multi-view modeling of

- Concurrency and timing, State charts, Hybrid systems, QoS and Resource Tradeoffs

Code Generator

Separation of Concerns



Multi-Dimensional Models

- Functional Models:
  - Data flow, control flow, state machines, task networks, Simulink, …
- Para-Functional Dimensions:
  - Timeliness
  - Concurrency
  - Fault-Tolerance
  - Security
  - Power Management
  - QoS
  - Modality
  - Energy
- Hardware Model
- Deployment
  - Software to processor binding
  - Middleware
  - RTOS
  - Communications

Verification proceeds along each dimension as well

Composition and decomposition aspects

Other Capabilities

- Automatic configuration and sizing of
  - Hardware modules
  - RTOS
  - Middleware components
  - …
- Guidelines and recommendations if configuration fails
  - Expertise captured in tool(s)

Understanding Failures

- Components, sub-systems, systems and systems of systems must have **error models**
  - What happens if a input parameter is bad?
  - What happens if an output is out of range?
  - How does the error propagate?
  - How does it impact the stability and correct operation of the physical plant?
- **Layered** and **hierarchical error models**
  - *Not* **spaghetti error models**

Specifying Requirements

- **Need**: Software Engineering?
  - Or Systems Engineering?
- Is it a software problem or a "system" problem?
- Requirements capture and requirements tracking?
- Specify what can be done
  - not do what can be specified

Core Research Challenges

- Bringing '-ilities' together
- Comprehensive modeling frameworks
- QoS and multi-resource management
  - Energy management
- Practical verification techniques
- Automation

An Analytical Framework for QoS Mgmt

- Across a wide variety of applications:
  - Cell-phones, telephone systems, video-conferencing, radar systems, VoIP, home media networks, …
- The system consists of multiple applications
  - An application is characterized by
    - **Multiple QoS dimensions**
      - Latency, encryption level, availability, …
      - Video frame-rate, pixel resolution, window size, audio sampling rate, …
    - **Multiple resource requirements**
      - CPU cycles, disk bandwidth, network bandwidth
    - Utility curves express satisfaction along each QoS dimension
- Maximize global utility by
  - Appropriately apportioning system resources to applications

Challenges for "Utility"-based Approaches

- Known Results
  - Optimal solutions for simpler cases (single-resource, independent QoS dimensions)
  - **Near-optimal solutions (in near real-time)** for complex cases
    - Multiple resources and multiple QoS dimensions
- Can we define a framework with supporting tools to determine application-level utilities for various QoS operating points?
  - Inter-application weights can be customizable by end-users
- Can we build tools to automatically determine resource needs for various QoS operating points?

Energy Management

- Given a pot of energy E, how to maximize the mission-time of the system satisfying operational and resource parameters?
- Given a pot of energy E, how to maximize the mission-time of a system and maximizing the QoS delivered to the users?
- Given that a certain weight W yields E, how to maximize battery time, maximize (1/W) and maximize QoS?
- How does one deal with dynamic changes?
- How does one model future energy demands?

Core Research Challenges

- Bringing '-ilities' together
- Comprehensive modeling frameworks
- QoS and multi-resource management
  - Energy management
- Practical verification techniques
- Automation

Verification and Validation

- Verification
  - Need serious, practical and hybrid advances in theorem-proving and model checking
  - Composability of logical properties will be critical
- Test and validation tools
  - Auto-vector generation
  - "Hardware in the loop" simulation for control of interaction with physical plants

Core Research Challenges

- Bringing '-ilities' together
- Comprehensive modeling frameworks
- QoS and multi-resource management
  - Energy management
- Practical verification techniques
- Automation

Tools and Automation

The Semi-Conductor Revolution

- Modeling, analysis and simulation tools
- Design *vs* Logic Implementation *vs* "Fabs"
- Systems of Systems

Cell Libraries FPGAs ASICs

Blades Motherboards I/O Cards

Chassis "Boxes"

The Software Revolution?

- System Models
- Models
- Components
- Functions/objects

- Domain experience is key: automotive, avionics (defense), software radios, …
  - GUI design has long been doable by "models".

Other Challenges

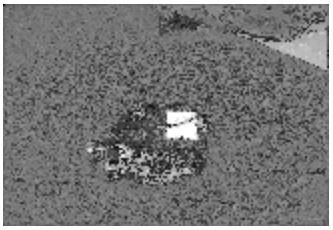
Non Ad-Hoc Sensor Networks

- Like the internet
  - Flexible, extensible, reconfigurable, odd structures but essentially with an underlying structure
    - Core routers, edge routers, subnets, end-points
  - Things can change but not continually
- Monitor health and operation of physical environments
  - Factories and equipment, buildings, campuses, homes, bridges, …
- Maximize mission time, availability, resistance to intrusion, …
- Self-reconfigurability but remote viewability and controllability

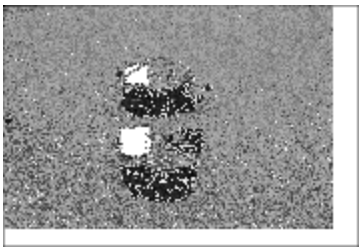
Cooperating Entities

- "Star Wars" robot armies
  - Make local decisions but cooperate and coordinate
  - Within an ***uncertain*** environment
- What are the theory and principles of operation for high probability of success?
- What guarantees failure?
- What kinds of uncertainties can and cannot be dealt with?

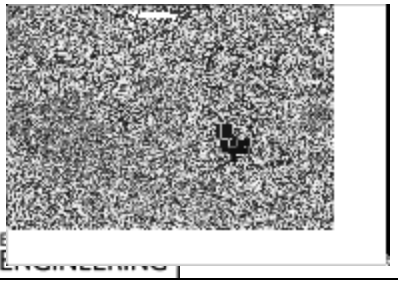
Let's "drive" a Robot



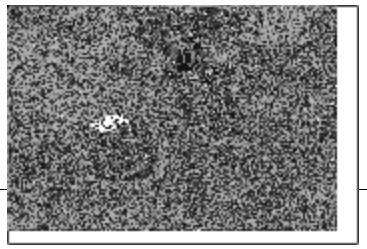
Let's drive a *Virtual* Robot



"Robot: Just avoid obstacles"



"Virtual Robot: Do the same"



Missing Technology Pieces

Legend:
Scientific challenges
Engineering challenges

- Theory
  - Unifying theory for satisfying multiple -ilities
- Systems
  - Secure RTOS and Middleware
- QoS Frameworks
  - Quantifying Utility (subjective and objective)
- Coordination and cooperation
  - in uncertain environments
- Automation
  - LOTS of tools operating at high levels of abstraction
- Verification
  - Practical advances in theorem -proving and model checking
- Test and validation tools
  - Auto-vector generation
- Standardized modeling environments
  - Integrating framework where all the pieces come together!

Concluding Remarks

- Need unifying theory (RT+FT+Sec)
  - Supported by a host of automation tools
    - Analysis, simulation and instrumentation
- Modeling framework to capture functional and para-functional attributes separately
  - "Divide and Conquer"
  - Large repositories of components to choose from
    - Each component is customizable…
  - New software gets written as components only
  - "Optimizing code-generators" to auto-generate efficient code for a given platform