EMBEDDED SYSTEMS

© → atomtracker.tk, 2017
EMBEDDED SYSTEMS MARKET

>USD 62 BILLION BY 2023

~5.3% ANNUAL GROWTH RATE

~12% FOR AUTOMOTIVE ELECTRONICS

EUROPE EMBEDDED MARKET SIZE (USD BILLION)

Est. GDP of Serbia 2016

© → Global Markets Research, 2016
REAL-TIME SYSTEMS

**COMPUTERS, I/O DEVICES + SOFTWARE**
- Intensive interaction with external environment
- Time-dependent variations in the state of the external environment
- Need to keep control over all individual parts of the external environment and to react to changes

**TIMELINESS**
- Reactivity
- Accuracy
- Duration
- Completion
- Responsiveness

**TIMING CONSTRAINTS**
- Operational correctness does not solely depend on the logical result but also on the time at which the result is produced
- The computed response has an application-specific utility function
- Correctness is defined in the value domain and in the time domain
- A logically-correct response produced later than due may be as bad as a wrong response

© → Prof. Tullio Vardanega, University of Padova
APPLICATION REQUIREMENTS

RELIABILITY
Measured in terms of maximum acceptable probability of failure
Typically in the range $10^{-10}$ to $10^{-5}$ per unit of life/service time

SAFETY-CRITICAL SYSTEMS
E.g. Airbus A-320: $10^{-10}$ probability of failure per hour of flight
One failure in $10^{10}$ hours of flight (about 11.5 million years!)

MISSION-CRITICAL SYSTEMS
E.g., satellite system: between $10^{-6}$ and $10^{-7}$ probability of failure per hour of operation
One failure in $10^7$ hours of operation (about 11,306 years!)
KEY CHARACTERISTICS

REAL-TIME SYSTEMS ARE

Dependable (proven)

Complexity
• Algorithmic, mostly because of the need to apply discrete control over analog and continuous physical phenomena
• Development, mostly owing to more demanding verification and validation processes

Heterogeneity of components and of processing activities

Extreme variability in size and scope

REAL-TIME SYSTEMS MUST HAVE

Temporal predictability
• Need for static (off-line) verification of correct temporal behavior
• Not easy at all
Response to events triggered by the external environment as well as by the passing of time
• Double nature: event-driven and clock- (or time-) driven

Continuity of operation
Software architecture is inherently concurrent
MEETING REAL-TIME REQUIREMENTS

SYSTEM-LEVEL PREDICTABILITY

Given a set of demanding real-time requirements and an implementation based on fast HW and SW, how can one show that those requirements are met?

- Surely not only via testing and simulation
- Maiden flight of space shuttle, 12 April 1981: 1/67 probability that a transient overload occurs during initialization; and it actually did!

IT IS NOT SUFFICIENT TO MINIMIZE THE AVERAGE RESPONSE TIME OF APPLICATION TASKS

"REAL-TIME COMPUTING IS NOT EQUIVALENT TO FAST COMPUTING“ [STANKOVIC, 88]
TIMING ANALYSIS (TA)

TIMING ANALYSIS INPUTS
A software controlling some process
A hardware platform executing software
Required reaction time of that software

TIMELINESS GUARANTEES
Worst-case execution time (WCET)
Best-case execution time (BCET)

©→ Casse, Sainrat 2010
EXECUTION TIME VARIES A LOT

ARCHITECTURES ARE MORE COMPLEX

Average case improving features
Caches, pipelines, branch prediction, speculation
Worst case trustworthy, but pessimistic

MPC755: \( x = a + b \)

© Casse, Sainrat 2010

© Wilhelm et. al. 2009
WCET

WCET COMPUTATION
For any input data and all initial logical states
• So that all execution paths are covered
For any hardware state
• So that worst-case conditions are in effect

TIMING ANALYSIS TYPES
Static
• Abstract model of the HW and of the program
Measurement-based
• On the real HW or a cycle-accurate simulator
• The high-watermark value can be $\leq$ WCET
Hybrid

© → Mezzeti, 2014
WCET (CONTD.)

TRIGGERING THE WCET BY TEST IS DIFFICULT

Worst-case **input covering** all executions of a real program is intractable in practice

Worst-case **initial state** is difficult to determine with modern HW

EXACT WCET NOT GENERALLY COMPUTABLE

A WCET estimate or **bound** are key to predictability

- Must be **safe** to be an upper bound to all possible executions
- Must be **tight** to avoid costly over-dimensioning
STATIC TIMING ANALYSIS IN DETAIL

Legend:
- Data
- Phase

© Wilhelm et. al. 2009
OPEN ISSUES

CAN WE ALWAYS TRUST HW MODELING?
HOW MUCH OVERESTIMATION DO WE INCUR?
• INCLUSION OF INFEASIBLE PATHS
• OVERESTIMATION INTRINSIC IN ABSTRACT STATE COMPUTATION

SAFENESS IS AT RISK

When local worst case does not always lead to global worst case

When timing anomalies occur due to complex hardware architectures (e.g., out-of-order pipelines)
Even improper design choices (e.g., cache replacement policies)
  Counter-intuitive timing behavior
Faster execution of a single instruction causes long-term negative effects
  Very difficult to account for in static analysis

WEAKNESSES OF USER ANNOTATIONS
• LABOR INTENSIVE AND ERROR PRONE
A GLIMPSE INTO THE FUTURE OF TA

STATIC DETERMINISTIC TIMING ANALYSIS (SDTA)
Well established and trusted
Based on sound mathematical abstractions
Likely to introduce pessimism
Risk of inaccurate timing models or annotations
Best fit: Simple HW and simple SW

MEASUREMENT-BASED PROBABILISTIC TIMING ANALYSIS (MBPTA)
Cost effective approach
Based on extreme value theory
Not fully mature
Worst-case path coverage
Best fit: Arbitrary complex systems with proper HW support
TRENDS IN CRITICAL REAL-TIME INDUSTRY

ADD NEW FUNCTIONALITY
Systems become more complex
- Autonomous drive, collision avoidance, etc.
Need more computing power
- E.g. 4x faster collision avoidance → UAV flying indoors

CONTAIN VERIFICATION COST
Industry wants incremental verification
- Enabled by using robust partitioning among applications
WHAT MANY-CORES HAVE TO OFFER?

CO-HOSTING SEVERAL APPLICATIONS

IMPROVE PERFORMANCE WITH THREAD LEVEL PARALLELISM (TLP)

Better performance per watt
Simple core design

Tilera TILEPro64  Kalray MPPA  Intel SCC
MANY-CORES ARE NOT A SILVER BULLET!

REAL-TIME INDUSTRY IS STILL LEARNING HOW TO EXPLOIT THEM

Harder to analyze w.r.t. single cores
- Inter-task interference
- Techniques developed for single-core do not work

M. Fernandez et. al. @EMSOFT 2012
WHAT’S WRONG WITH MULTICORES

ACCESES TO SHARED RESOURCES
Network on chip, shared caches, memory, I/O, etc.
Inter-task interference

WHAT ABOUT PARALLEL APPLICATIONS?
Critical sections
Barriers
Synchronization primitives

SO FAR WAY TO GO ➔ TIME COMPOSABILITY
WCET estimates must hold for any workload
Increases pessimism
COMPOSITIONALITY

SYSTEMS ARE BUILT FROM BUILDING BLOCKS
Software components
Hardware subsystems

CAN WE ANALYZE IT IN PIECES?
Reduce complexity of the analyses
Reduce margin between WCET estimate and real WCET

ISSUES
Decomposition of the system
Combining the results of partial analyses
COMPOSITIONAL ANALYSIS

TA CONSIDERS ALL POSSIBLE INTERACTION

WCET estimation must consider the worst system integration

\[ WCET_{\text{integration}} = WCET_{\text{analysis\_isolation}} \]

TA CONSIDERS ONLY SOME INTERFERENCE

The rest is considered at system integration as \( \Delta \)

\[ WCET_{\text{integration}} = WCET_{\text{analysis\_isolation}} + \Delta \]
TIME PREDICTABLE HARDWARE

GRP₁

GRP₂

GRP₃

GRP₄

GRP₁

GRP₂

GRP₃

GRP₄

MANY-CORES IN AVIONICS

WCET IMPROVEMENT

3DPP

Stereo Navigation

M. Panic et al. @EMSOFT 2014
Hierarchical libraries and other facilities that support large-scale software development.

Strong compile-time type checking.

Safe object-oriented programming facilities.

Language-level support for concurrent programming.

A coherent approach to real-time systems development.

High-performance implementations.

Well-defined subsetting mechanisms, and in particular the SPARK subset for formal verification.
USE CASES OF ADA

SAFETY-CRITICAL SYSTEMS

Avionics and air-traffic control
• Fly-by-wire on Boeing 777
• Honeywell Air Transport System

Space rockets
• Ariane 4 and 5

Train signaling systems
• TGV in France
• Metro in Paris, London, Hong Kong, etc.
ADA FEATURES

LANGUAGE FEATURES
Strong typing
Modularity mechanisms (packages)
Parallel processing (tasks, synchronous message passing, protected objects, and nondeterministic select statements), Exception handling
Generics.
Object-oriented programming (ADA95)
Pragmas

RUNTIME CHECKS
Access to unallocated memory
Buffer overflow errors
Range violations
Off-by-one errors
Array access errors
Other detectable bugs

MEMORY MANAGEMENT
Dynamic
Type safe
No generic or pointers without type
Allocation/deallocation through declared access types
Semantically supports garbage collection, though it’s usually not implemented
RAVENSCAR

ENFORCED BY MEANS OF A CONFIGURATION PRAGMA

PRAGMA PROFILE (RAVENSCAR);

Equivalent to a set of Ada restrictions plus three additional configuration pragmas
pragma Task_Dispatching_Policy (FIFO_Within_Priorities);
pragma Locking_Policy (Ceiling_Locking);
pragma Detect_Blocking;
RAVENCAR RESTRICTIONS

<table>
<thead>
<tr>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>No_Abort_Statements</td>
</tr>
<tr>
<td>No_Dynamic_Attachment</td>
</tr>
<tr>
<td>No_Dynamic_Priorities</td>
</tr>
<tr>
<td>No_Implicit_Heap_Allocations</td>
</tr>
<tr>
<td>No_Local.Protected_Objects</td>
</tr>
<tr>
<td>No_Local_Timing_Events</td>
</tr>
<tr>
<td>No_Protected_Type_Allocators</td>
</tr>
<tr>
<td>No_Relative_Delay</td>
</tr>
<tr>
<td>No_Requeue_Statements</td>
</tr>
<tr>
<td>No_Select_Statements</td>
</tr>
<tr>
<td>No_Specific_Termination_Handlers</td>
</tr>
<tr>
<td>No_Task_Allocators</td>
</tr>
<tr>
<td>No_Task_Hierarchy</td>
</tr>
<tr>
<td>No_Task_Termination</td>
</tr>
<tr>
<td>Simple_Barriers</td>
</tr>
<tr>
<td>Max_Entry_Queue_Length =&gt; 1</td>
</tr>
<tr>
<td>Max_Protected_Entries =&gt; 1</td>
</tr>
<tr>
<td>Max_Task_Entries =&gt; 0</td>
</tr>
<tr>
<td>No_Dependence =&gt; Ada.Asynchronous_Task_Control</td>
</tr>
<tr>
<td>No_Dependence =&gt; Ada.Calendar</td>
</tr>
<tr>
<td>No_Dependence =&gt; Ada.Execution_Time.Group_Budget</td>
</tr>
<tr>
<td>No_Dependence =&gt; Ada.Execution_Time.Timers</td>
</tr>
<tr>
<td>No_Dependence =&gt; Ada.Task_Attributes</td>
</tr>
</tbody>
</table>
Thank you for your attention!

MILOS PANIC
SENIOR SOFTWARE DEVELOPER

+381 11 440 79 84
milos.panic@endava.com