Timing Analysis: New Directions After the Breakthrough

Andreas Ermedahl, PhD
andreas.ermedahl@mdh.se
Mälardalen Real-Time Research Center (MRTC)
Sweden
What breakthrough?

★ Timing analysis is now mature enough to be a standard component in embedded system development
★ Examples:
  - Avionics software
  - Software controlling in-vehicle communication networks
  - Real-time operating system code
  - Space applications
★ Timing analysis research has developed into companies
  - AbsInt (static analysis)
  - Tidorum (static analysis)
  - Rapita Systems (measurements)
What new directions?

» The technique is mature, but some challenges still remain

- As indicated by Prof. Wilhelm’s speech

« This speech will:

- Present current ARTIST2 activities to overcome some of these challenges
- Present some of the remaining challenges for timing analysis research

« But first, a short repetition of the ingredients of timing analysis
Definition of WCET

★ A program might have many executions
- Each with a different execution time

★ Worst Case Execution Time = WCET
- The longest execution time of the program
- Safe (and tight) estimates are needed
- One task in isolation

★ Other measures: BCET, ACET
Obtaining WCET estimates

★ Measurement
- Industrial practice
- Add safety margin

★ Static analysis
- Research front
- Safe estimates
Static WCET analysis

★ Do not run the program – analyze it!
  ➤ Relying on models of the program and the hardware upon which it runs
★ Safe WCET estimates
  ➤ If the models and all inputs are correct

Measurements give times in this region
Static analyses give times in this region
Causes of timing variations

★ Characteristics of the program
- A program can often execute in several different ways
- Application characteristics
- Input data dependencies

★ Timing characteristics of the hardware
- Cache memories
- Pipelines
- ...

```c
int foo(int max)
{
    int i, j, total;
    i = 0;
    j = 1;
    while(i <= max)
    {
        if (j < 5)
            j++;
        if (j > max)
            break;
        total = total + j - 2;
        i++;
    }
    return total;
}
```
WCET analysis ingredients

- **Program representation**
  - Code, control-flow graphs, ...

- **Flow analysis**
  - Determine possible execution paths through the program

- **Low level analysis**
  - Determine execution time for program parts on hardware

- **Calculation**
  - Combine flow info and low-level times to generate a WCET estimate
int foo(int max)
{
    int i, j, total;
    i = 0;
    j = 1;
    while(i <= max)
    {
        if(j < 5)
            j++;
        if(j > max)
            break;
        total = total + j - 2;
        i++;
    }
    return total;
}
WCET & program analysis

* Timing analysis = program analysis
  - Derive run-time properties of the program

* Many type of program analyses exists

* Analyses have different quality and complexity
  - Eg. timing for program parts can be derived both by static analysis or by measurements

* One can give valueable input to another
  - Eg. value ranges for variables useful to derive loop bounds as well as addresses for data accesses
    - Different memory areas might have different access times

* WCET research groups often specialized in some particular types of analyses
ARTIST2 & WCET

**The ARTIST2 Timing Analysis cluster**
- Includes most European WCET analysis researchers
- Includes WCET analysis tool vendors

**Objectives:**
- Integrate European efforts on Timing Analysis of Real-Time Systems
- Combine the best components of existing European Timing-Analysis tools and prototypes
- Thereby preserving the existing lead of European Research and Industry in this important sector
**ARTIST2 Timing Analysis**

- **Goal:** A standard WCET analysis tool platform with well-defined interfaces
  - Allowing different WCET tools to cooperate and exchange results
  - Easier comparison of research results
  - Easier porting/development of WCET analyses

- **Modes of cooperation**
  - Meetings, research visits, management structure, financing, ...

- **The resulting platform will be used in teaching the technology all over Europe**
  - Industry seminars, tutorials, etc.
Example: The CRL2 Format

- Control-Flow Representation Language, 2nd version
  - Format already used in aiT tool
  - Designed to support various analyses and hardware architectures

- Starting point for interface format

- Control flow described by native syntax elements
  - Graphs, routines, blocks, edges, …

- Analysis results are stored in attributes
  - One exchange format for the whole tool chain

- Attributes are distinguished by contexts
  - Precision matches that of analyses
ARTIST2 cluster activities

* Work on the Timing-Analysis platform
  - Definition of tool platform and interfaces (2005)
  - Supporting and documenting the interfaces (2005)
  - Implementation and evaluation (2005)
  - Demonstrator (2006)

* Work on increased timing predictability
  - Jointly with Execution-Platforms cluster

* Integrating WCET with program compilation
  - Jointly with Compilation cluster

* See: http://artist.cs.uni-sb.de for more information
New Directions for WCET

The ARTIST2 tool platform will define
- How to represent the program
- How to express various analysis results

It does not specify how attributes should be derived
- Many different type of analyses can be used

Challenges remains
- Some listed in the following slides

```c
void foo(int j, int a[])
{
    int i;
    for(i=100, i>0; i--)
    {
        if(j>50)
            a[i] = 2 * i;
        else
            a[i] = i;
    }
}
```
Flow Analysis Research

★ Not yet fully possible to automate the WCET analysis on a ’one-click’ basis
   ▶ Detailed system knowledge often required
   ▶ Manual annotations to constrain program flow
   ▶ Often parametrical WCET due to input data dependencies

★ Experiences from industrial case studies

★ Flow analysis for WCET focus in ongoing research project at MRTC
   ▶ Using abstract interpretation
   ▶ Syntactical analysis

Easy to derive loop bound

for(i=0; i<100; i++) {...}

Harder to derive loop bound

for(p=list->begin(); p!=list->end(); p++) { ... }
**WCET & Generated Code**

- Much embedded system code generated by higher-level modeling and design tools
  - Esterel, Ascet, Matlab, Scade, ...
- The resulting code structure depends on code generator
  - Often simpler than handwritten code
- Should be possible to integrate with WCET analysis tools
  - The analysis can be automated
  - Less user interaction required
  - Eg. loop bounds can be provided directly by the modeling tool
Hardware model derivation

Static WCET analysis relies on models of the hardware timing
- Many embedded processors exist
- New models are required when porting WCET analysis to new hardware

How to derive such models?
- Hardware manuals not detailed enough
- Hardware manuals might contain bugs
- Details are manufacturer secrets

How to verify the derived models?
- Much systematic testing required
- How to obtain 100% accuracy?

Ongoing work in German AVACS research center
- Derivation of timing models from formal hardware specifications (VHDL abstract timing model)
Compiler interaction

- Today – commercial WCET analysis tools analyses binaries
- Another possibility – interaction with the compiler
  - Easier to identify variables and to understand what the program is intended to do
- There exists many compilers for embedded systems
  - Very fragmented market
  - Each specialized on a few particular targets
  - Targeting code size and execution speed
- Integration with WCET analysis tools opens new possibilities:
  - Compile for timing predictability
  - Compile for reduced WCET
**WCET Analysis in Education**

★ Goal: Make students aware of static WCET analysis tools
★ Example: Porting the Bound-T WCET analysis tool to the Renesas H8/300 processor
★ Renesas H8/300 part of Lego Mindstorms
   - Used in RT courses in academia
★ Planned for release during next two months
   - Will be used in RT courses at MRTC and Uppsala
   - Will be available for other universities as well
Timing Predictability

- Precision of timing analysis depends on predictability properties of:
  - The *overall system architecture*, i.e. the partitioning of code, hardware, tasks, ...
  - The *computer architecture*, i.e. processor architecture, peripherals, bus protocol
  - The *software development process*, e.g. design discipline, programming language, coding guidelines
  - The *operating systems characteristics*, e.g. scheduling discipline

- **Design and code for Timing Predictability**
  - WCET analysis is a key component
Conclusions

* Timing analysis is mature enough to be a standard part in the embedded system programmer’s tool chest
  - The ARTIST2 timing analysis cluster provides a foundation for continued development in this direction
  - Interesting research challenges remains 😊
The End!
Configurable hardware

- Configure your processor to suit your application
  - Statically
  - Dynamically
- Put parts of your code into application specific instruction sets
- WCET analyzer has to be adapted to each application
  - Use information available from synthesis tool