Introduction to Worst-Case Execution-Time Analysis

Peter Puschner
How do we know the timing is right?
Time in RTS Construction

**Design**
Architecture, resource planning, schedules

**Implementation**

**Timing Analysis**
Schedulability analysis, WCET analysis
From Design to Implementation

Task set with precedence constraints and deadline

Task sequence: execution times, response time

Can we guarantee that: response time < deadline?
In general it is infeasible to model all possible execution scenarios and combinations of task execution times.

Timing analysis abstracts the different execution times of each task to one single value $\Rightarrow$ WCET.
Schedulability and WCET Analysis

Schedulability objects
- Units of execution (simple tasks) with WCET
- Precedence relations
- Synchronization, communication, mutual exclusion
- Priorities

WCET-analysis objects
- Simple tasks

Interference ... (nasty and therefore widely neglected)
- “external” changes of task state that influence exec. time
Simple Task

- Inputs available at start
- Outputs ready at the end
- No blocking inside
- No synchronization or communication inside
- Execution time variations only due to differences in
  - inputs
  - task state at start time
  (no external disturbances)
Why not just Measure WCET?

- Start Timing Measurement
- Execute Task on Target HW
- Stop Timing Measurement

Timer, Logic Analyzer, etc.

WCET estimate?
Why not just Measure WCET? (2)

- Measuring all different execution traces of a real-size program is intractable in practice. (e.g., mid-size task: $10^{40}$ different paths)

- Selected test data for measurement may fail to trigger the longest execution trace.

Rare execution scenarios may be missed (e.g., exception handling, ...).
Static WCET Analysis

Static WCET Analysis: computes upper bounds for the execution time of pieces of code

Execution time: time it takes to execute
- a given piece of code (without disturbance)
- in a given application context (inputs, state)
- on a given machine

Cave: execution time is not response time
Goals for WCET Analysis

WCET analysis = modeling
Complexity necessitates modeling tradeoffs

- WCET bounds must be safe
- WCET bounds should be tight
- WCET analysis cost should be reasonable
WCET Analysis – Quality

frequency

BCET  WCET  WCET Bound

t
WCET Determinants

- Possible sequences of actions of the code (= execution paths) in given application [→ code, context]
- The duration of each occurrence of an action on each possible path [→ hardware, state]
Exec. Time: Simple vs. Complex Arch.

Simple Architecture Model

Duration of each action $a_i$ is constant:

$$xt(p_k) = \sum n_{k,i} t(a_i)$$

- $p_k$ ... path $k$
- $t(a_i)$ ... WCET of action $a_i$
- $n_{k,i}$ ... number of times action $a_i$ is executed on path $p_k$

Complex Architecture Model

Durations of actions $a_i$ vary depending on history:

$$xt(p_k) = \sum t(a_{i,j(k)})$$

- $t(a_{i,j(k)})$ ... WCET of the $j^{th}$ action $a_i$ executed on path $k$

Reasons: pipelining, caches, parallelism, …
WCET Analysis Problems

Path analysis: identifying (in)feasible paths to obtain flow facts

- Syntactic restrictions – code
- Semantic restrictions – code
- Input-data space – application
- Automatic analysis vs. user annotations

Modelling of hardware timing

- CPU architecture
- Implementation of instructions
- Resource limitations and state dependencies
Representation Levels

Source-Code Representation

- Extraction/annotation of possible execution paths – flow facts

Machine-Language Representation

- Translation of flow facts
- Execution-time analysis, computation of WCET bound
Representation Levels

Flow-fact mapping between representation levels has to be formally correct
→ support by the code transformer
otherwise mapping may be ambiguous.

Possible support by code transformer:

- emit execution history
  (flow facts update by an external tool)
- perform flow facts transformation in parallel to code transformation.
Generic WCET Analysis Framework

- Source code
- Compilation
  - Transformation of Flow Facts
  - Extraction of Flow Facts
  - Exec-Time Modeling
- Object code
- Calculation of Execution Scenarios

WCET
Representation Level

WCET analysis of Matlab/Simulink models:
Summary

WCET definition
- Code; context (application, situation); machine

WCET versus WCET modelling

Problems of the analysis
- Flow fact analysis and annotation
- Hardware modeling
- Bridging the representation levels