



T-CREST



Time-predictable Multi-Core Architecture for Embedded Systems

Compiler and WCET Analysis Tool Chain

Gernot Gebhard, AbsInt Angewandte Informatik GmbH
Daniel Prokesch, Vienna University of Technology

Outline

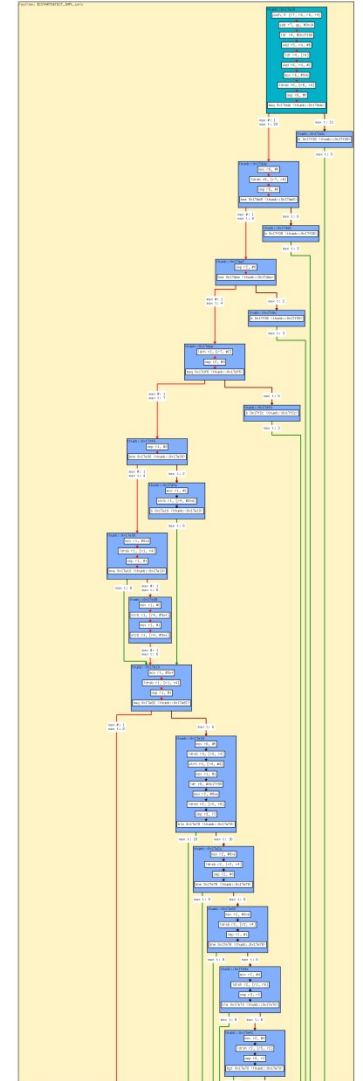
- Goals of T-CREST
- WCET Analysis
 - ◆ The Timing Problem
 - ◆ Timing Analysis Framework
- Compilation Tool Chain
 - ◆ Compiler Structure
 - ◆ Integrating Compiler and WCET Analysis
- T-CREST Contributions
- Demo

Goals of T-CREST

- Timing-Predictable Patmos Architecture
- Safe WCET Bounds Mandatory
 - ◆ Static Timing Analysis
- Generation of Code with Predictable Timing
 - ◆ Low Timing Variability
 - ◆ Good Worst-Case Performance
- Information Exchange between Compiler and WCET Analysis

Embedded Control Software

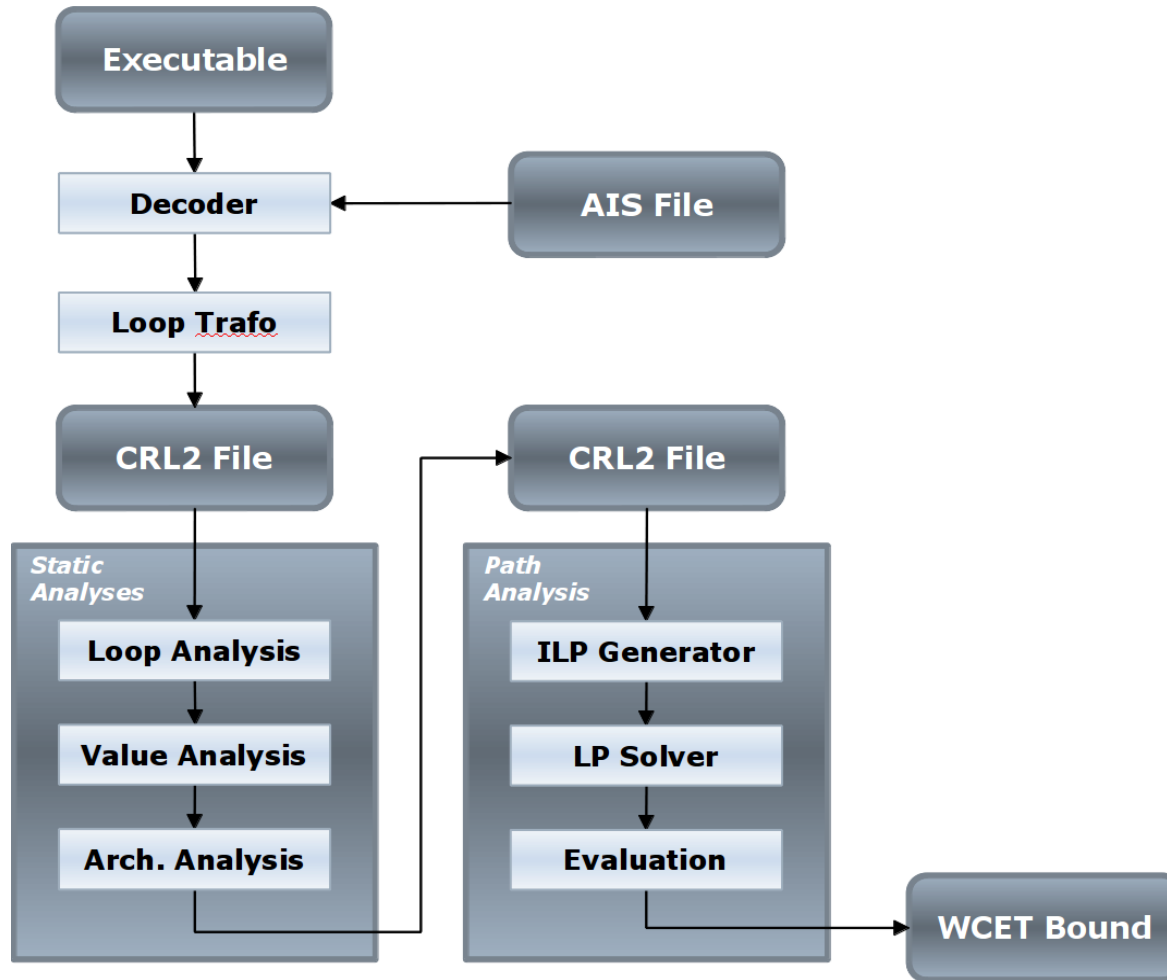
- Large and Complex Structure
- Large Functionality
- Code-Generating Tools
- 3rd Party Software
 - ◆ RTOS (e.g., RTEMS)
 - ◆ Communication (e.g., CAN)



aiT WCET Analyzer

- Solution to the Timing Problem
- Abstract Interpretation
 - ◆ Loop/Value Analysis
 - ◆ Architectural Analysis (Caches/Pipeline)
- Integer Linear Program
 - ◆ Path Analysis
- Combined in an intuitive GUI

WCET Analyzer Framework



Information Exchange

■ Annotation for Flow Facts

◆ Loop Bound Example:

```
loop "main.L1" bound: 4;
```

■ Annotations for Call/Branch Targets

◆ Switch Table Example:

```
instruction "main" -> 48 byte branches to:
```

```
    "main" -> 60 byte,
```

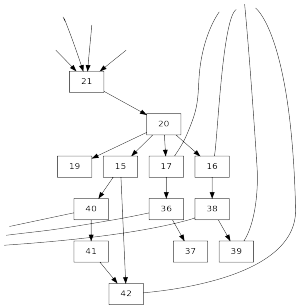
```
    "main" -> 74 byte,
```

```
    "main" -> 80 byte;
```

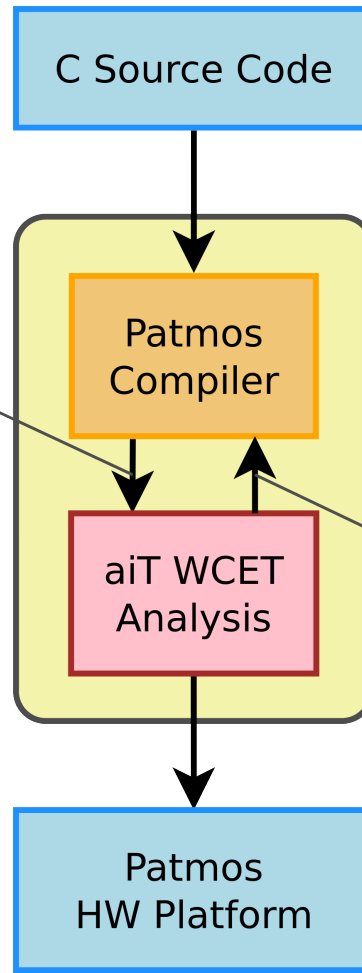
■ Decoder benefits from Compiler Integration

Tasks of the Compiler

Preserve information available during compilation process



```
...
loop ".LBB2_1" max 71;
loop ".LBB2_2" max 2;
loop ".LBB2_26" max 14;
loop ".LBB2_23" max 4;
...
```

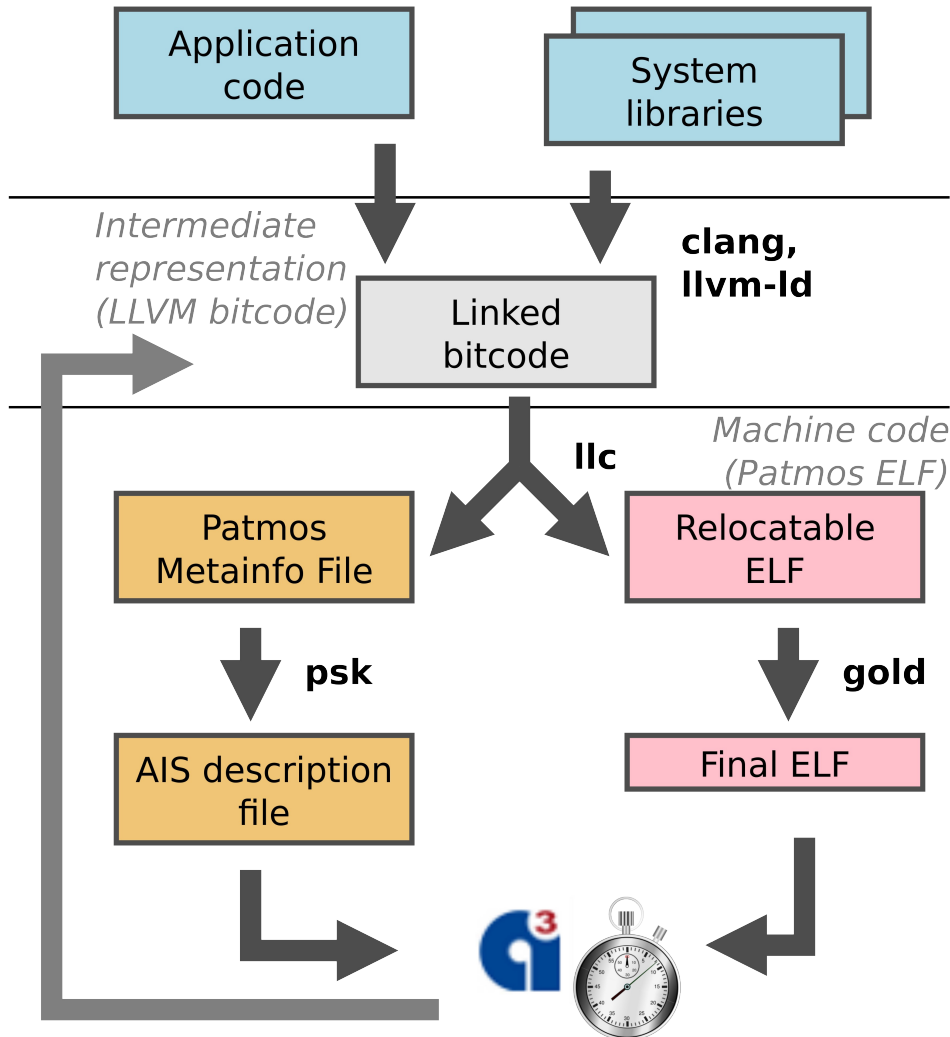


Feedback from WCET analysis for WCET-guided optimization



Compiler Overview

Source code (C language)



- Based on LLVM Compiler Framework
- C source files translated into LLVM bitcode (clang)
- Bitcode files (user, libraries) linked together to a single bitcode file
- Translated to machine code (relocatable ELF)
- Object code linker for final relocations
- Code generator exports information for WCET analysis

Support for Patmos Features

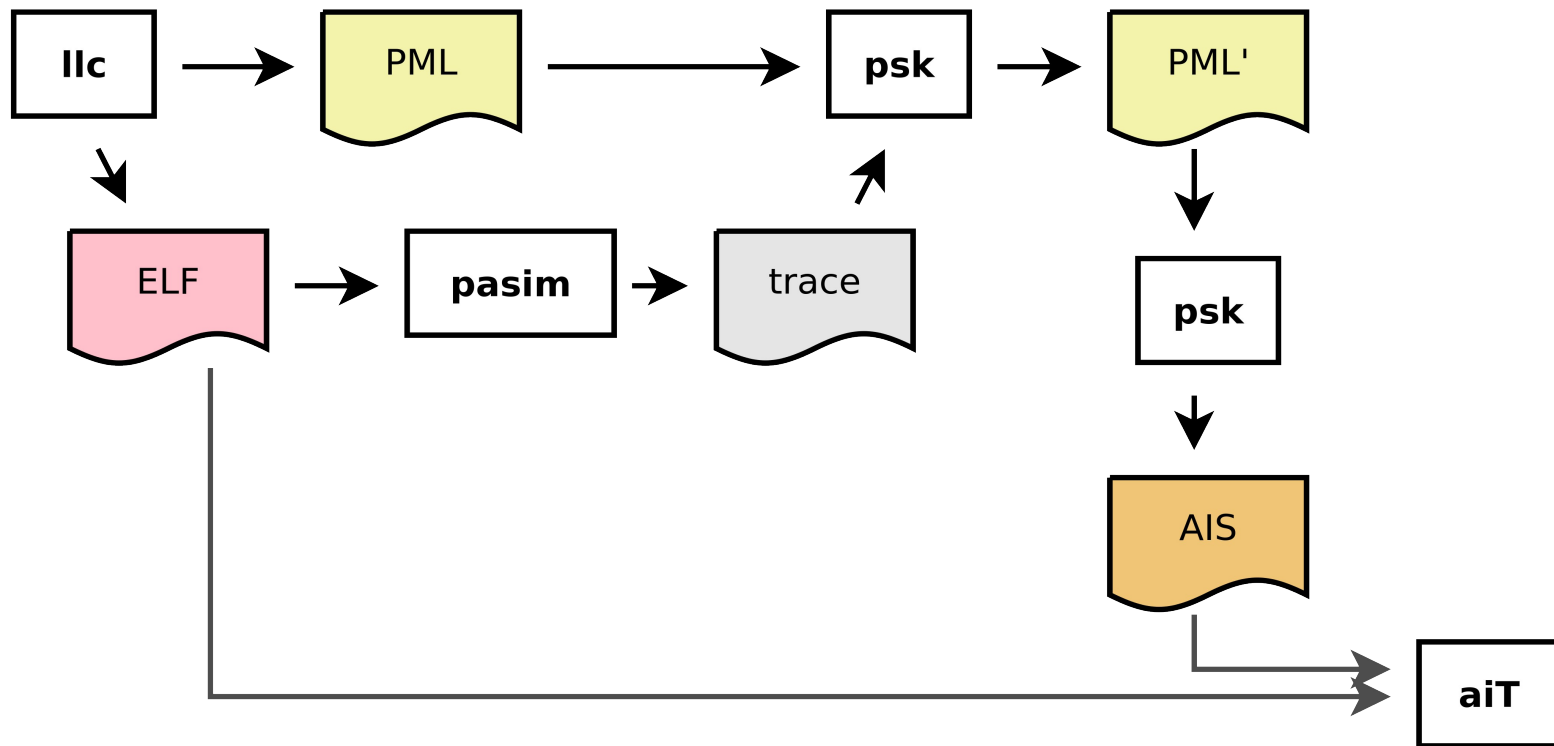
- Support for the Time-Predictable Architectural Specifics
 - ◆ Method Cache
 - ◆ Stack Cache
 - ◆ Typed Memory Operations
 - ◆ Predictable Pipeline
 - ◆ Predicated Execution

Integration Compiler – WCET Analysis

- PSK (Patmos Swiss Army Knife) Toolset
 - ◆ PML (Patmos Metainfo Language) file format
 - ◆ Extend, merge, visualize, im-/export
- Exchange information
 - ◆ Between Compiler and WCET Analysis
- Enrich by information from external tools
 - ◆ e.g., Simulation

Example PSK Usage Scenario

- Annotate PML with simulation information to obtain preliminary loop bounds



T-CREST Contributions

- Integration of Compilation and Timing Analysis
- Compilation Techniques
 - ◆ Code with stable timing
 - ◆ WCET-driven optimization
 - ◆ Leverage Patmos Features for Predictability
- Detailed feedback about timing behaviour at source level

<http://github.com/t-crest/>

Demo

