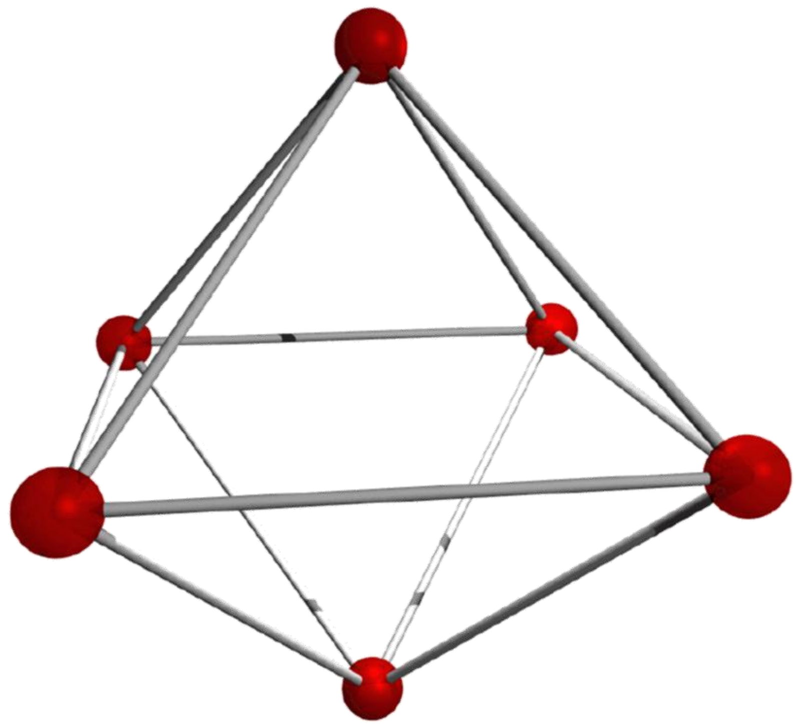
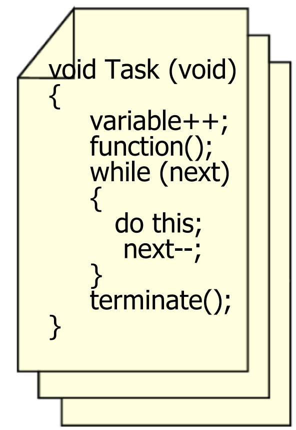
Worst-Case Timing Estimation and Architecture Exploration in Early Design Phases



AbsInt
Angewandte Informatik GmbH

Early Estimation Problem

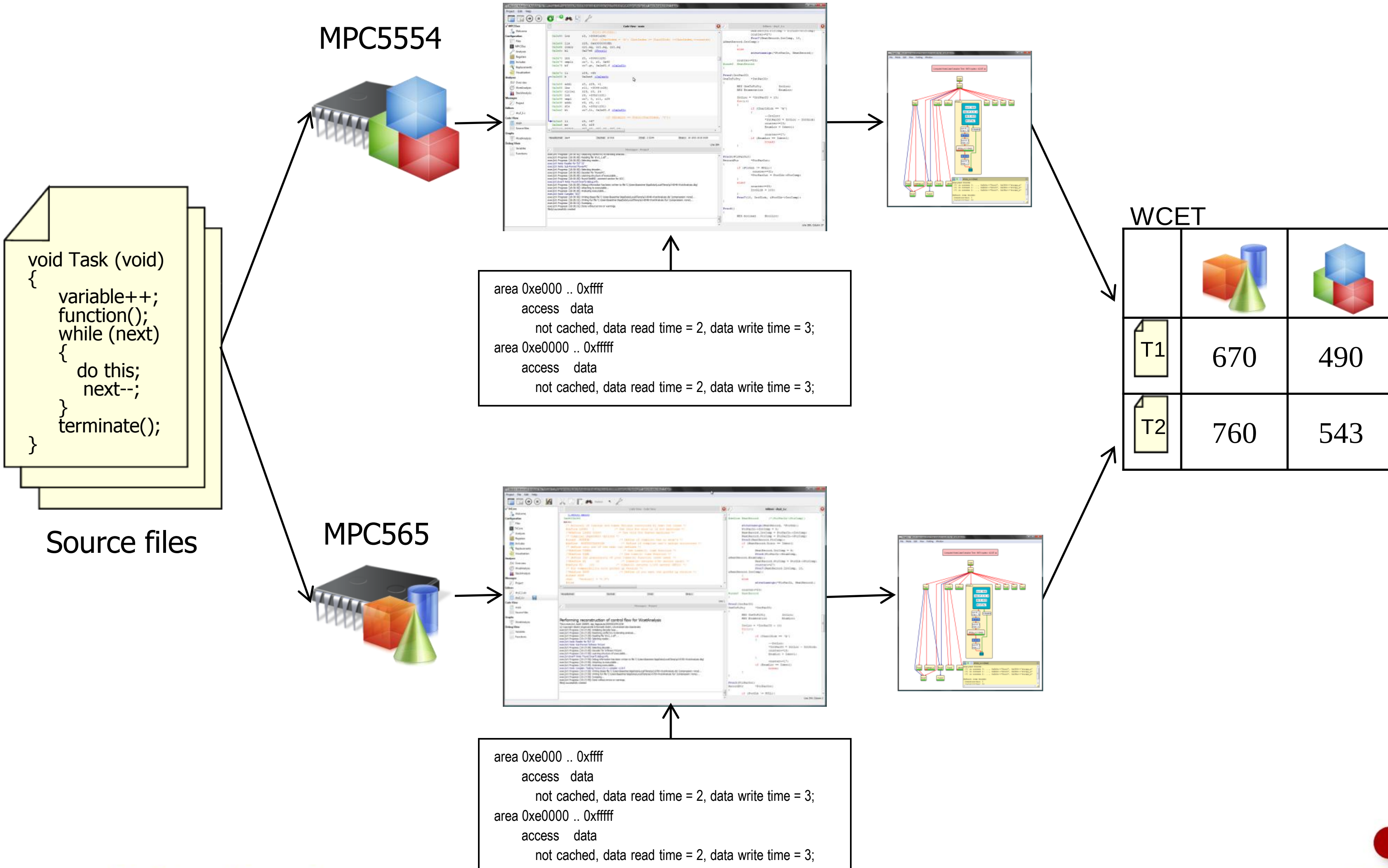


Source code or models

TimingExplorer

- Goal: assist in the exploration of alternative system configurations
- Design:
  - based on aiT
  - parameterizable core
- Requirements: (representative) source code

Core choice

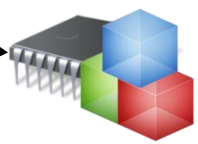


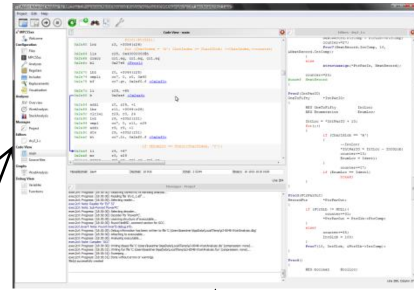
T1	670	490
T2	760	543

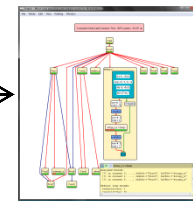
Architecture configuration

```
void Task (void)
{
      variable++;
      function();
      while (next)
      {
         do this;
          next--;
      }
      terminate();
}
```

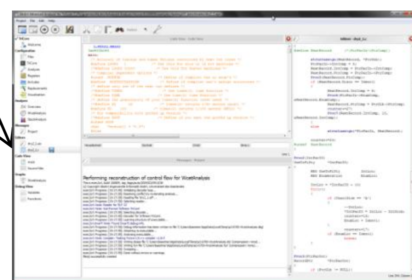
Source files

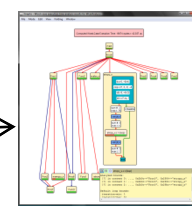






```
cache instruction
    set-count = 128, assoc = 4, line-size = 32,
    policy = LRU, may = empty
and data
    set-count = 128, assoc = 4, line-size = 32,
    policy = LRU, may = empty;
```





```
area 0xe000 .. 0xffff
     access   data
         not cached, data read time = 2, data write time = 3;
area 0xe0000 .. 0xfffff
     access    data
         not cached, data read time = 2, data write time = 3;
```

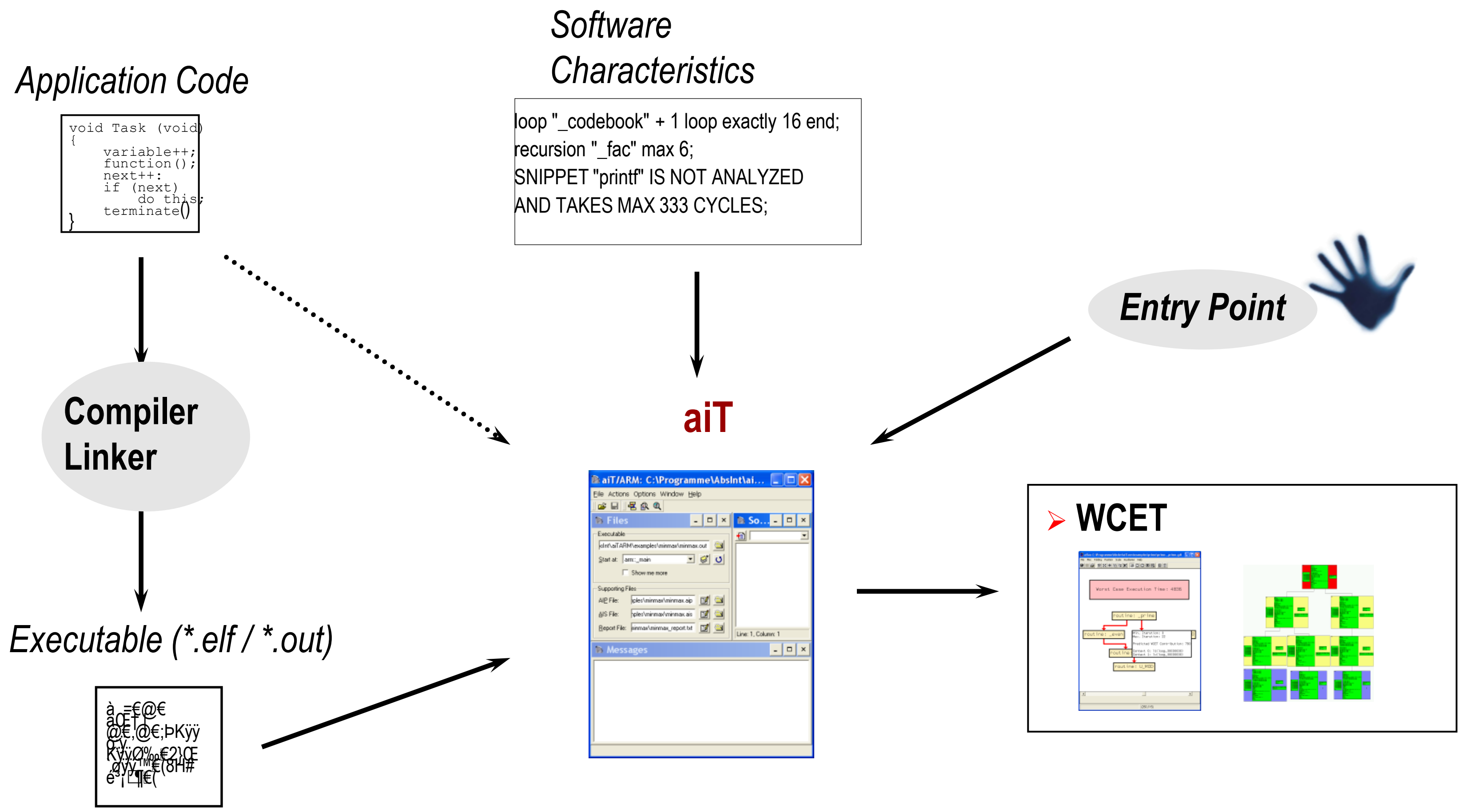
WCET

T1	490	650
T2	543	760

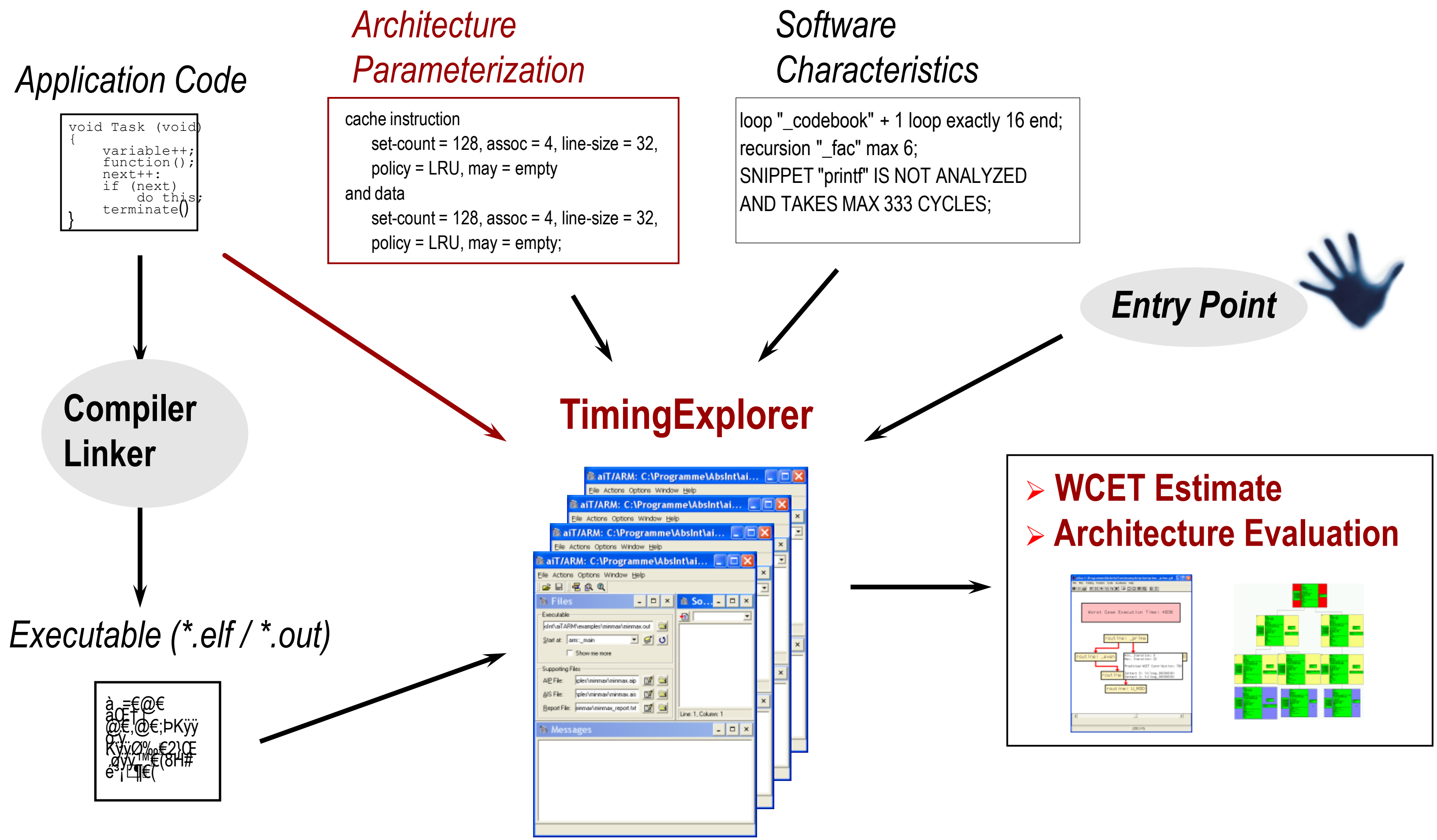
From Source to Executable

- Compilation and linking are user’s responsibility
- Compiler effects
  - Production compiler
  - Standard compiler

aiT


Application Code
void Task (void)
{
variable++;
function();
next++:
if (next)
do this;
terminate()
}
Software Characteristics
loop "_codebook" + 1 loop exactly 16 end;
recursion "_fac" max 6;
SNIPPET "printf" IS NOT ANALYZED
AND TAKES MAX 333 CYCLES;
Entry Point
Compiler Linker
aiT
Executable (*.elf / *.out)
WCET

TimingExplorer



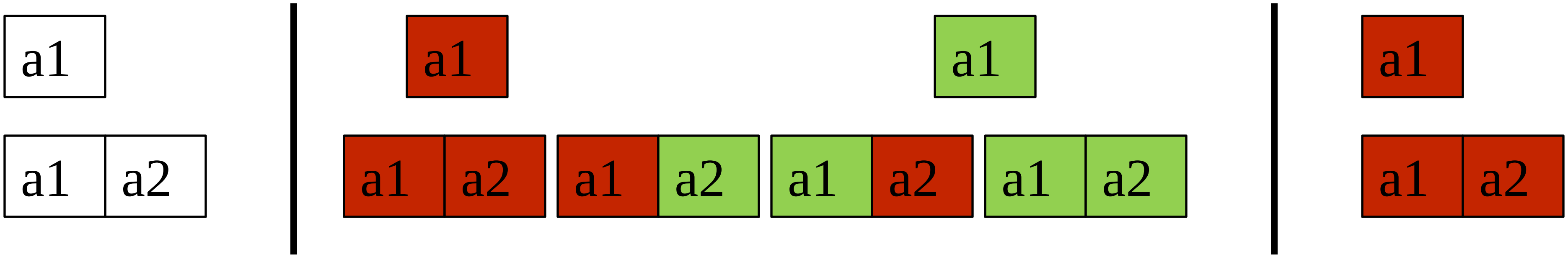
AbsInt
Angewandte Informatik

Architecture Specification

- Cache
  - Size, line size, replacement policy, associativity
  - Cache access behaviour:
    - Default strategy for unknown accesses – hit/miss
    - Turn off cache analysis and treat all accesses as hit/miss/unknown
- Completely parameterizable memory map

Pipeline Analysis

- Use local worst case



- Goal
  - Reduce computation resource usage
  - Speed up the analysis

Source-code Analyses

- Goal: reduce the need for manual annotation
- May introduce imprecision:

```
for (int i=0; i<100; i++)
    ...
```

- Analyses: function pointer resolution, loop bounds
  - Framework: SATIrE, LLNL-ROSE, PAG

Future work

- Case study
- Integration in a system-level analysis
  - XML Timing Cookies

Source-code Analyses

- Goal: reduce the need for manual annotation
- May introduce imprecision:

```
for (int i=0; i<100; i++)
    …
```

- Analyses: function pointer resolution, loop bounds
  - Framework: SATIrE, LLNL-ROSE, PAG

Function Pointer Resolution

- Compute a points-to set for each function pointer variable
- Handles complex structures
- Output: annotations

Test benchmark

Program	Version	Lines of code	Indirect calls
diction	0.7	2 037	3
grep	2.0	12 417	3
gzip	1.2.4	8 163	4
sim6890	0.1	3 290	97

Loop-bound Analysis

- Express the upper loop bound as a parametric formula
- One symbolic formula per loop
  - value analysis => concrete bounds

Loop-bound analysis results

- Detects and assigns a bound to all loops
- Precision (Mälardalen benchmark suite; provided precise results of the value analysis)
  - > SWEET
  - ≥ oRange
- Automatically generated code
- Effective for typical counter loops