**aiT Worst-Case Execution Time Analyzer**

**Timing Guarantees for Real-Time Systems**

aiT WCET Analyzer computes **tight bounds** for the worst-case execution time of tasks in safety-critical systems. These bounds are **safe**, i.e., they are valid for any input scenario and each task execution.

aiT is based on statically analyzing a task’s intrinsic **cache and pipeline behavior**, thus enabling the development of complex hard real-time systems on state-of-the-art hardware.

---

### Application Code

```c
void Task (void)
{
    variable++;
    function();
    next++;
    if (next)
        do_this();
    terminate();
}
```

### Specifications (*.ais)

- `clock kHz; 10200`
- `loop + loop exactly end;`
- `"_codebook" 16`
- `recursion max;`
- `"_fac" 6`
- `snippet is not analyzed and takes max cycles;`
- `"printf" 333`
- `flow + bytes / + bytes is max;`
- `"U_MOD" "U_MOD" 0xAC 0xC4 4`
- `area from to is readonly;`
- `0x20 0x497`

---

### The Challenge:

- **Measuring** the execution time of a task is typically **not safe**. It is often impossible to prove that all the conditions determining maximum execution time are taken into account. Code instrumentation and debug information change the timing behavior.

- Hardware speculation by caches, pipelines, etc. complicates the task of determining the WCET, since the execution time of a single instruction may depend on the **execution history**.

- Analysis methods that do not consider **cache and pipeline behavior** typically seriously **overestimate** the WCET, leading to a substantial waste of hardware resources.

---

### This is where aiT steps in:

- aiT-computed bounds are **valid for all inputs** and each execution of a task. Extensive timing testing is now a thing of the past.

- aiT directly analyzes binary executables. This means that **no modification of your tool chain** or the program’s operational behavior and performance is required.

- aiT-computed bounds are tight and reflect the **real performance** of your system. Cache and pipeline effects are fully taken into account. Ensuring deadline adherence is no longer done at the expense of hardware resources.
Why do you need aiT?

The worst-case execution time (WCET) of each task in a real-time system has to be known prior to its execution. In event-triggered or periodic systems (e.g. RMA), the WCET is required for schedulability analysis; in time-triggered systems (e.g. TTA, FlexRay, ...), it is required for determining a static schedule.

The increasing performance of microcontrollers enables more and more functionality to be implemented by a single embedded control unit. The software is complex and the timing behavior of the interacting software components rarely known. Typically it is not practical – or even possible – to test the system with all potential inputs.

aiT computes safe upper bounds on the WCET of each task, providing full data and control coverage, enabling timing safety.

aiT Features:

- **Visualization** of the analysis results providing detailed information about key timing aspects, e.g. the worst-case path or the machine state at any given program point.
- Various statistics, interactive tables, graphs and charts that let you quickly identify bottlenecks and other areas of interest.
- Analysis report files for documentation and certification purposes, as well as for integration with numerous software development tools.
- Qualification Support Kits are available providing support for automatic tool qualification up to the highest criticality levels (DO-178B, DO-178C, ISO26262, IEC61508, EN50128).

**Key Benefits:**

- aiT can replace error-prone methods based on testing and measuring, enhancing safety.
- aiT has been qualified as a verification tool according to various safety norms, including DO-178B/C for Level A software, enabling certification of safety-critical real-time software.
- aiT provides automatic tool support for calculating the WCET of your applications, saving development time.
- aiT safely determines the timing behavior of interacting software components, enabling software integration.