Schedulability & WCET Analysis
Task Scheduling

utilization of CPU

\[ P(i), [E(i), L(i)], \ldots : \text{period or earliest/latest arrival or } \ldots \text{ for } T_i \]

\[ C(i): \text{execution time for } T_i \]

\[ D(i): \text{deadline for } T_i \]

Scheduler

\{ T_4, T_1, T_3 \} \text{ ready ordered according to some given priority: (e.g. Fixed Priority, Earliest Deadline, \ldots)}
**Classical Scheduling Theory**

### Utilisation-Based Analysis

- A simple sufficient but not necessary schedulability test exists

\[
U \equiv \sum_{i=1}^{N} \frac{C_i}{T_i} \leq N \left(2^{1/N} - 1\right)
\]

\[U \leq 0.69 \text{ as } N \to \infty\]

Where \(C\) is WCET and \(T\) is period

### Response Time Equation

\[R_i = C_i + \sum_{j \in \text{hpt}(i)} \left[ \frac{R_j}{T_j} \right] C_j\]

Where \(\text{hpt}(i)\) is the set of tasks with priority higher than task \(i\)

Solve by forming a recurrence relationship:

\[w_i^{n+1} = C_i + \sum_{j \in \text{hpt}(i)} \left[ \frac{w_j^n}{T_j} \right] C_j\]

The set of values \(w_i^n, w_i^{n+1}, w_i^{n+2}, \ldots\) is monotonically non decreasing

When \(w_i^n = w_i^{n+1}\) the solution to the equation has been found, \(w_i^n\) must not be greater than \(R_i (\text{e.g. } 0 \text{ or } C_i)\)

- **Simple to perform**
- Overly conservative
- Limited settings
- Single-processor

**ARTIST Design PhD School, Beijing, 2011**

**Kim Larsen [3]**
Modeling Task

Scheduler

ARTIST Design PhD School, Beijing, 2011
Kim Larsen [4]
Modeling Scheduler

Scheduler

T

T

T

T

ready

done

Stop

Run

ARTIST Design PhD School, Beijing, 2011

Kim Larsen [5]
Modeling Queue

In UPPAAL 4.0

User Defined Function

```c
// Put an element at the end of the queue
void enqueue(id_t element)
{
    int tmp=0;
    list[len++] = element;
    if (len>0)
    {
        int i=len-1;
        while (i>1 && P[list[i]]>P[list[i-1]])
        {
            tmp = list[i-1];
            list[i-1] = list[i];
            list[i] = tmp;
            i--;
        }
    }
}

// Remove the front element of the queue
void dequeue()
{
    .......
```
Schedulability = Safety Property

May be extended with preemption

\( \neg (\text{Task0.Error or Task1.Error or …}) \)

ARTIST Design PhD School, Beijing, 2011

Kim Larsen [7]
Dealing with Resources

bool resource[N];

bool available(id_t id)
{
    return !resource[need[id]];
}

void take(id_t id)
{
    assert(!resource[need[id]]);
    resource[need[id]] = true;
}

void release(id_t id)
{
    assert(resource[need[id]]);
    resource[need[id]] = false;
}
Preemption – Stopwatches!

Scheduler

Task

Defeating undecidability 😊
Handling realistic applications?

Smart phone:

[Application from Marcus Schmitz, TU Linkoping]

Jan Madsen
Aske Brekling
Michael R. Hansen/ DTU
Timed Automata for a task
Smart phone

- Tasks: 114
- Deadlines: [0.02 : 0.5] sec
- Execution: [52 : 266.687] cycles
- Platform:
  - 6 processors, 25 MHz
  - 1 bus

Verified in 1.5 hours!

ARTIST Design PhD School, Beijing, 2011

Kim Larsen [13]
ESA Missions

- Solar System, cold dust clouds and cores, star and galaxy formations, cataloging galaxies, gravitational lensing, cosmic microwave background, topology of the universe...

**TERMA**

- Terma: Develop software for Attitude and Orbit Control System

*ARTIST Design PhD School, Beijing, 2011*  
*Kim Larsen [14]*
Herschel & Planck Satellites

- **Application software (ASW)**
  - built and tested by Terma:
  - does attitude and orbit control, telecommanding, fault detection isolation and recovery.

- **Basic software (BSW)**
  - low level communication and scheduling periodic events.

- **Real-time operating system (RTEMS)**
  - Priority Ceiling for ASW,
  - Priority Inheritance for BSW

- **Hardware**
  - single processor, a few buses, sensors and actuators

**Requirements:**

Software tasks should be schedulable.
CPU utilization should not exceed 50% load
UPPAAL Model

- One template for CPU scheduler:
  - maintains a queue of ready tasks
  - schedules tasks with highest priority in the queue
  - reschedule if higher priority task arrives to the queue
- One template per each ASW task.
- Two templates for BSW tasks: 1 plain, 1 using resource.
- One template for “idle” task to count CPU utilization.
- WCET is modeled by stopwatches and lower bound.
- WCRT is modeled by stopwatches.
- Deadline is enforced as guard on WCRT.
- System is schedulable if no deadline violated.
- CPU load is \( \frac{\text{used time}}{\text{total time}} \).
Fig. 11. Gantt chart of a schedule from the first cycle: green means ready, blue means running, cyan means suspended, red means blocked. R stand for resources: CPU_R=0, Icb_R=1, Sgm_R=2, PmReq_R=3, Other_RCS=4, Other_SF1=5, Other_SF2=6.
<table>
<thead>
<tr>
<th>ID</th>
<th>Task</th>
<th>Specification</th>
<th>Blocking times</th>
<th>WCRT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Period</td>
<td>WCET</td>
<td>Deadline</td>
</tr>
<tr>
<td>1</td>
<td>RTEMS_RTC</td>
<td>10.000</td>
<td>0.013</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>AswSync_SyncPulseIsr</td>
<td>250.000</td>
<td>0.070</td>
<td>1.000</td>
</tr>
<tr>
<td>3</td>
<td>Hk_SamplerIsr</td>
<td>125.000</td>
<td>0.070</td>
<td>1.000</td>
</tr>
<tr>
<td>4</td>
<td>SwCyc_CycStartIsr</td>
<td>250.000</td>
<td>0.200</td>
<td>1.000</td>
</tr>
<tr>
<td>5</td>
<td>SwCyc_CycEndIsr</td>
<td>250.000</td>
<td>0.100</td>
<td>1.000</td>
</tr>
<tr>
<td>6</td>
<td>Rt1553_Isr</td>
<td>15.625</td>
<td>0.070</td>
<td>1.000</td>
</tr>
<tr>
<td>7</td>
<td>Bc1553_Isr</td>
<td>20.000</td>
<td>0.070</td>
<td>1.000</td>
</tr>
<tr>
<td>8</td>
<td>Spw_Isr</td>
<td>39.000</td>
<td>0.070</td>
<td>2.000</td>
</tr>
<tr>
<td>9</td>
<td>Obdh_Isr</td>
<td>250.000</td>
<td>0.070</td>
<td>2.000</td>
</tr>
<tr>
<td>10</td>
<td>RtSdb_P_1</td>
<td>15.625</td>
<td>0.150</td>
<td>15.625</td>
</tr>
<tr>
<td>11</td>
<td>RtSdb_P_2</td>
<td>125.000</td>
<td>0.400</td>
<td>15.625</td>
</tr>
<tr>
<td>12</td>
<td>RtSdb_P_3</td>
<td>250.000</td>
<td>0.170</td>
<td>15.625</td>
</tr>
<tr>
<td>14</td>
<td>FdirEvents</td>
<td>250.000</td>
<td>5.000</td>
<td>230.220</td>
</tr>
<tr>
<td>15</td>
<td>NominalEvents_1</td>
<td>250.000</td>
<td>0.720</td>
<td>230.220</td>
</tr>
<tr>
<td>16</td>
<td>MainCycle</td>
<td>250.000</td>
<td>0.400</td>
<td>230.220</td>
</tr>
<tr>
<td>17</td>
<td>HkSampler_P_2</td>
<td>125.000</td>
<td>0.500</td>
<td>62.500</td>
</tr>
<tr>
<td>18</td>
<td>HkSampler_P_1</td>
<td>250.000</td>
<td>6.000</td>
<td>62.500</td>
</tr>
<tr>
<td>19</td>
<td>Acb_P</td>
<td>250.000</td>
<td>6.000</td>
<td>50.000</td>
</tr>
<tr>
<td>20</td>
<td>IoCyc_P</td>
<td>250.000</td>
<td>3.000</td>
<td>50.000</td>
</tr>
<tr>
<td>21</td>
<td>PrimaryF</td>
<td>250.000</td>
<td>34.050</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>RCSControlF</td>
<td>250.000</td>
<td>4.070</td>
<td>239.600</td>
</tr>
<tr>
<td>23</td>
<td>Obf_P</td>
<td>100.000</td>
<td>1.100</td>
<td>100.000</td>
</tr>
<tr>
<td>24</td>
<td>Hk_P</td>
<td>250.000</td>
<td>2.750</td>
<td>250.000</td>
</tr>
<tr>
<td>25</td>
<td>StsMon_P</td>
<td>250.000</td>
<td>3.300</td>
<td>125.000</td>
</tr>
<tr>
<td>26</td>
<td>TmGen_P</td>
<td>250.000</td>
<td>4.860</td>
<td>250.000</td>
</tr>
<tr>
<td>27</td>
<td>Sgm_P</td>
<td>250.000</td>
<td>4.020</td>
<td>250.000</td>
</tr>
<tr>
<td>28</td>
<td>TcRouter_P</td>
<td>250.000</td>
<td>0.500</td>
<td>250.000</td>
</tr>
<tr>
<td>29</td>
<td>Cmd_P</td>
<td>250.000</td>
<td>14.000</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>NominalEvents_2</td>
<td>250.000</td>
<td>1.780</td>
<td>230.220</td>
</tr>
<tr>
<td>31</td>
<td>SecondaryF_1</td>
<td>250.000</td>
<td>20.960</td>
<td>189.600</td>
</tr>
<tr>
<td>32</td>
<td>SecondaryF_2</td>
<td>250.000</td>
<td>39.690</td>
<td>230.220</td>
</tr>
<tr>
<td>33</td>
<td>Bkgnd_P</td>
<td>250.000</td>
<td>0.200</td>
<td>250.000</td>
</tr>
</tbody>
</table>
## Effort and Utilization

### Table: Uppaal resources vs Herschel CPU utilization

<table>
<thead>
<tr>
<th>cycle limit</th>
<th>Uppaal resources</th>
<th>Herschel CPU utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU, s Mem, KB</td>
<td>Idle, μs Used, μs Global, μs Sum, μs Used, %</td>
</tr>
<tr>
<td>1</td>
<td>465.2 60288 173456</td>
<td>91225 160015 250000 251240 0.640060</td>
</tr>
<tr>
<td>2</td>
<td>470.1 59536 174234</td>
<td>182380 318790 500000 501170 0.637580</td>
</tr>
<tr>
<td>3</td>
<td>461.0 58656 175228</td>
<td>273535 477705 750000 751240 0.636940</td>
</tr>
<tr>
<td>4</td>
<td>474.5 58792 176266</td>
<td>363590 636480 1000000 1000700 0.636480</td>
</tr>
<tr>
<td>6</td>
<td>474.6 58796 178432</td>
<td>545900 955270 1500000 1501170 0.636847</td>
</tr>
<tr>
<td>8</td>
<td>912.3 58856 352365</td>
<td>727110 1272960 2000000 2000700 0.636480</td>
</tr>
<tr>
<td>13</td>
<td>507.7 58796 186091</td>
<td>1181855 2069385 3250000 3251240 0.636734</td>
</tr>
<tr>
<td>16</td>
<td>1759.0 58728 704551</td>
<td>1454220 2545850 4000000 4000070 0.636463</td>
</tr>
<tr>
<td>26</td>
<td>541.9 58112 200364</td>
<td>2363640 4137530 6500000 6501170 0.636543</td>
</tr>
<tr>
<td>32</td>
<td>3484.0 75520 1408943</td>
<td>2908370 5091700 8000000 8000070 0.636463</td>
</tr>
<tr>
<td>39</td>
<td>583.5 74568 214657</td>
<td>3545425 6205745 9750000 9751170 0.636487</td>
</tr>
<tr>
<td>64</td>
<td>7030.0 91776 2817704</td>
<td>5816740 10183330 16000000 16000070 0.636458</td>
</tr>
<tr>
<td>78</td>
<td>652.2 74768 257582</td>
<td>7089680 12411420 19500000 19501100 0.636483</td>
</tr>
<tr>
<td>128</td>
<td>14149.4 141448 5635227</td>
<td>11633480 20366590 32000000 32000070 0.636456</td>
</tr>
<tr>
<td>156</td>
<td>789.4 91204 343402</td>
<td>14178260 24821740 39000000 39000000 0.636455</td>
</tr>
<tr>
<td>256</td>
<td>23219.4 224440 11270279</td>
<td>23266890 40733180 64000000 64000070 0.636456</td>
</tr>
<tr>
<td>312</td>
<td>1824.6 124892 686788</td>
<td>28356520 49643480 78000000 78000000 0.636455</td>
</tr>
<tr>
<td>512</td>
<td>49202.2 390428 22540388</td>
<td>46533780 81466290 128000000 128000070 0.636455</td>
</tr>
<tr>
<td>624</td>
<td>3734.7 207728 1373560</td>
<td>56713040 99286960 156000000 156000000 0.636455</td>
</tr>
</tbody>
</table>
Safety Critical Java
SARTS: Schedulability Analysis

With
Bent Thomsen, Anders P. Ravn,
Thomas Bøgholm, Henrik Kragh-Hansen,
Petur Olsen, Rene R. Hansen,
Lone Leth Thomsen, Hans Søndergaard,
Java Object – Regional IKT Korridor

- Productivity of a programmer is increased with up to 700% by changing from C/C++ to Java!
- Number of well-educated Java programmers increasing!
- Java for hard real-time systems?
- Java and C/Assembler legacy code?
- Emerging new profiles and hardware implementations!
- Eclipse framework!

- Center for Embedded Software Systems
- Vitus Bering Denmark
- Polycom (Kirk Telecom A/S)
- Wirtek A/S
- Mechatronic Brick ApS
- Aalborg Industries A/S
- Prevas A/S
- Teknologisk Institut
- Tekkva Consult (project coordinator).

ARTIST Design PhD School, Beijing, 2011

Kim Larsen [22]
A Safety Critical System
Hardware

- JOP (Java Optimized Processor)
- Native execution of Java Bytecode
- Bytecode implemented in Microcode
- Avoid unpredictable data-cache
- Time predictable
- Developed new method and stack cache
- Implemented in FPGA
Java Optimizing Processor

Martin Schöberl
University of Tech., Vienna

ARTIST Design PhD School, Beijing, 2011

Kim Larsen [25]
public static void main(String[] args) {
    new SporadicPushMotor(
        new SporadicParameters(4, 4000, 60), 0);
    new SporadicPushMotor(
        new SporadicParameters(2, 4000, 60), 1);

    PeriodicMotorSpooler motorSpooler =
        new PeriodicMotorSpooler(
            new PeriodicParameters(4000));

    new PeriodicReadSensor(
        new PeriodicParameters(2000),
        motorSpooler);

    IMPLEMENTATIONS of SC Java
    On JOP and Ajile aJ-100
    Use existing schedulers and threads
    On Mechatronic Brick and Polycom (Kirk)
    Currently experimenting with JamVM

    Min interarrival
    Deadline

    Periodic Threads
    Sporadic Threads
    RunTimeSystem
    Relative Time
    Immortal and Raw Memory
    Preemptive FP Scheduling
    Priority Ceiling
SARTS – Overview

ARTIST Design PhD School, Beijing, 2011
Kim Larsen [27]
SARTS – TA Templates

- Translation of Basic Blocks into states and transitions
- Patterns for:
  - Loops
  - Monitor statements
  - If statements
  - Method invoke
  - Sporadic task release

ARTIST Design PhD School, Beijing, 2011

Kim Larsen [29]
Byte code – Timed Automata

```java
protected boolean run()
    if i<5 {
        i = i + 4;
    } else {
        i = i * 4;
    }
    return true;
```

Timing = WCET from microcode
public static void main(String[] args) {
    new SporadicPushMotor(
        new SporadicParameters(4, 4000, 60), 0);
    new SporadicPushMotor(
        new SporadicParameters(2, 4000, 60), 1);
    PeriodicMotorSpooler motorSpooler =
        new PeriodicMotorSpooler(
            new PeriodicParameters(4000));
    new PeriodicReadSensor(
        new PeriodicParameters(2000), motorSpooler);
    RealtimeSystem.start();
}

private void handleBrick() {
    Sensors.synchronizedReadSensors();
    int input = (Sensors.getBufferedSensor(0) + Sensors
                  .getBufferedSensor(1)) >> 1;
    if (input > lastRead) {
        lastRead = input;
    } else if ((lastRead - input) >= TRESHOLD) {
        awaitingBrick = false;
        if (lastRead > BRICK_DETECTED) {
            brickFound(lastRead);
        }
    }
}
SARTS – to Timed Automata

Detection of Deadline Violation
Integrated SARTS w ECLIPSE
Visualize WCET in ECLIPSE

18 methods + 4 tasks = 76 components

ARTIST Design PhD School, Beijing, 2011
Kim Larsen [32]
SARTS – Experiments

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Verification time</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Javac</td>
<td>14m 29s</td>
<td>Satisfied</td>
</tr>
<tr>
<td>Eclipse</td>
<td>1m 55s</td>
<td>Satisfied</td>
</tr>
</tbody>
</table>

Breadth First Search

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Verification time</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Javac</td>
<td>4m 23s</td>
<td>Satisfied</td>
</tr>
<tr>
<td>Eclipse</td>
<td>51s</td>
<td>Satisfied</td>
</tr>
</tbody>
</table>

Depth First Search + Agressive SS Reduction

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Verification time</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Javac</td>
<td>16s</td>
<td>Satisfied</td>
</tr>
<tr>
<td>Eclipse</td>
<td>9s</td>
<td>Satisfied</td>
</tr>
</tbody>
</table>

Convex Hull Approximation

Javac

Eclipse
**WCET**: Worst Case Execution Time

In general: hard or impossible to predict.

Determine tight upper time bound instead.

**ARTIST Design PhD School, Beijing, 2011**

Kim Larsen [35]
METAMOC

Timed automata models for hardware components and process functions:

Abstract process model and value analysis

Abstract hardware model with caching and pipelining

WCET 42 cycles
Overview of METAMOC

Annotated executable

Pipeline (UPPAAL model)

Main memory (UPPAAL model)

Cache specifications

disassemble (objdump, Dissy)

generate (Assembly-to-UPPAAL)

Control Flow Graph (UPPAAL model)

combine

value analysis (WALi)

Complete model (UPPAAL model)

Caches (UPPAAL models)

model check (UPPAAL)

WCET
Value analysis in METAMOC

- Memory addresses needed for cache hit/miss predictions
- Registers used as base and offset for memory accesses
- Overapproximate possible register values
- METAMOC uses Weighted Push-Down Systems (WPDSs) for an inter-procedural, control-flow sensitive value analysis
- Weighted Automata Library (WALi) utilised

---

Modeling in UPPAAL
GUI for METAMOC

http://metamoc.martintoft.dk
Status

- Started out with ARM9 support
  - Five stage pipeline, instruction cache, data cache, simple main memory
- Demonstrated the method convincingly—we thought
- The WCET community: “It’s a case study. You haven’t demonstrated the method’s modularity.”
- Now:
  - Support for ARM7, ARM9 and ATMEL AVR 8-bit
  - ... with modest effort
- Accepted paper for WCET 2010, the 10th Int’l Workshop on Worst-Case Execution-Time Analysis
Experiments

- Evaluation using WCET benchmark programs from Mälardalen Real-Time Research Centre
  - Applicability
  - Performance
- Discarded a number of programs
  - Floating point operations handled by software routines
  - Dynamic jumps
  - Some programs do not compile
- 21 programs for ARM and 19 programs for AVR
- Manually annotated loop bounds
Experiments / Future

- Looking for WCET benchmark programs with reference WCETs
- A general, cycle-accurate hardware emulator, turning a hardware description and a program into a WCET

Relative improvement in WCET for ARM9.
Analysis times in minutes for AVR and ARM9.
Conclusion & Future

**CONCLUSION**
- Reduction of Verification Gap
- Simplicity
  - Programming languages
  - Processor architecture
  - Scheduling principles
- RT Model checking
  - Less pessimistic
  - Enables more complex settings
  - Ease of use

**FUTURE**
- Support for more platforms
- Combined Schedulability & WCET Analysis
- Loop bounds?
- More applications
- Improved Model Checker
  - 64 bit version
  - Distributed MC
  - Abstract Caches using Lattice Types
- [sarts.boegholm.dk](http://sarts.boegholm.dk)
- [metamoc.martintoft.dk](http://metamoc.martintoft.dk)
- [www.uppaal.com](http://www.uppaal.com)

ARTIST Design PhD School, Beijing, 2011

Kim Larsen [45]