# Scheduling for Time-Triggered Network Communication

Jan Kamieth

jan.kamieth@informatik.haw-hamburg.de
Hochschule für Angewandte Wissenschaften Hamburg

14. Juni 2012

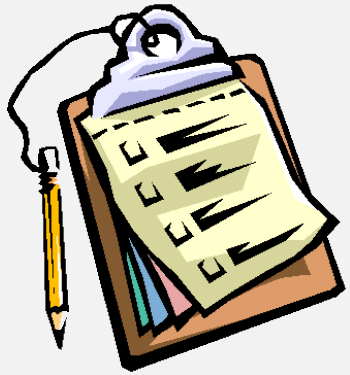
# Agenda

**(1) Rückblick**

(2) Verwandte Arbeiten

- Heuristiken
- Model Checking (SPIN)
- Model Checking (UPPAAL)

(3) Ausblick



# Rückblick: AW1

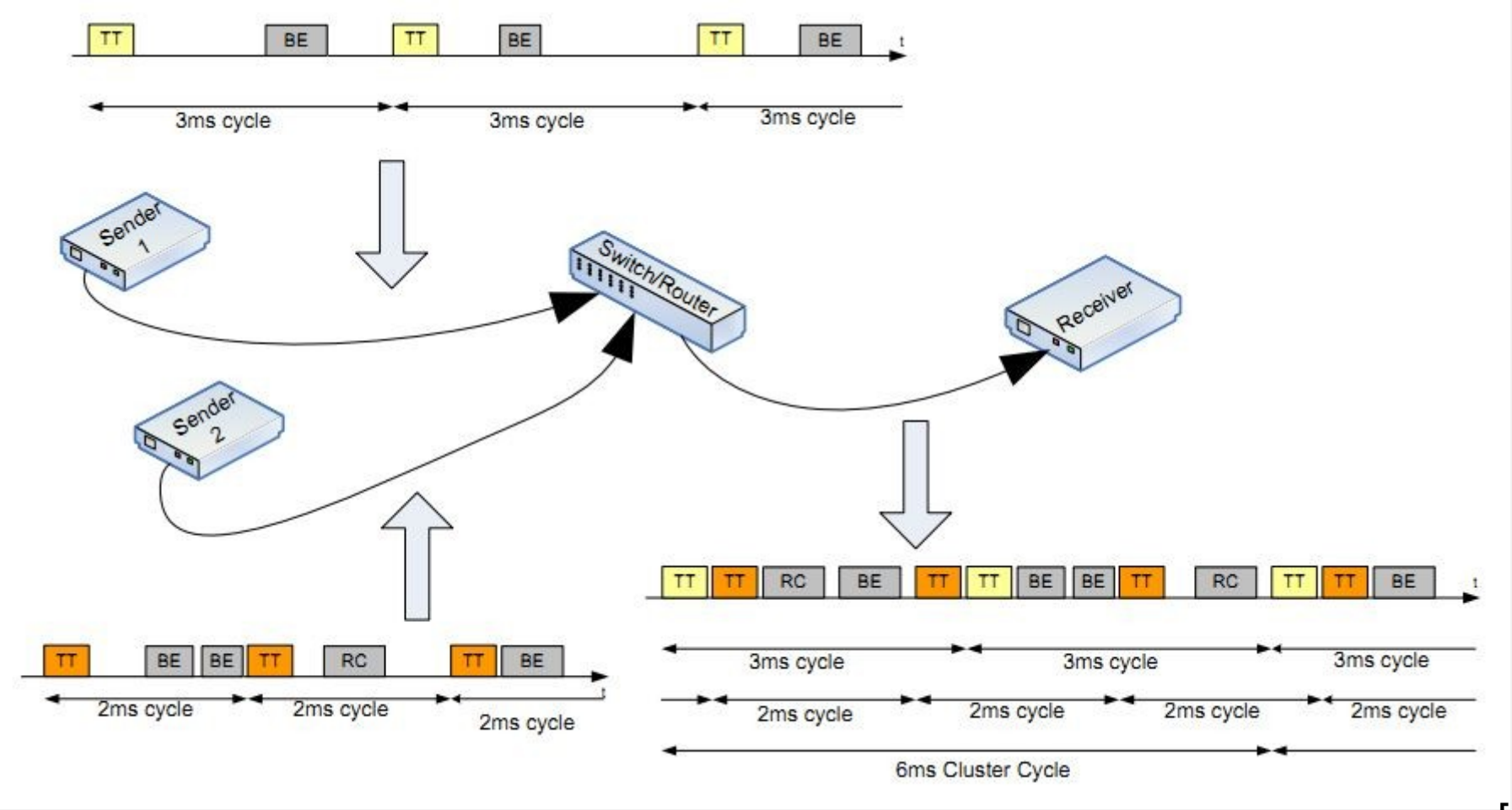
- Time-Triggered Ethernet (TTTech)
- Echtzeiterweiterung
- Switched bussystem

- Schedulingproblem: NP-vollständig
- Preemptive und non-preemptive
- Statisch und dynamisch

# Rückblick: TT-Ethernet

- **Time Triggered**
  - Zeitkritischer Netzwerkverkehr
  - Höchste Priorität
  - Minimum an Latenz und Jitter
- **Rate-constrained**
  - Ereignisbasierter Nachrichtentyp
  - Unterliegt keinem festen Scheduling
  - Werden von TT-Nachrichten verdrängt
- **Best-efford**
  - Standard Ethernet Netzwerkverkehr

# Rückblick: TTE-Schedule




[1]

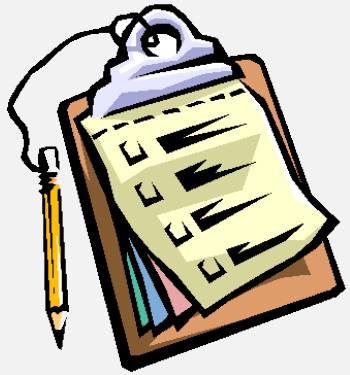
# Agenda

(1) Rückblick

**(2) Verwandte Arbeiten**

- **Heuristiken**
- Model Checking (SPIN)
- Model Checking (UPPAAL)

(3) Ausblick



# Heuristiken

## Scheduling with Bus Access Optimization for Distributed Embedded Systems

- Mehrere programmierbare und Hardware Prozessoren
- TTP Bussystem
- Netzwerk-Tasks als normale Prozess-Tasks
- Kontroll- und Datenabhängigkeiten

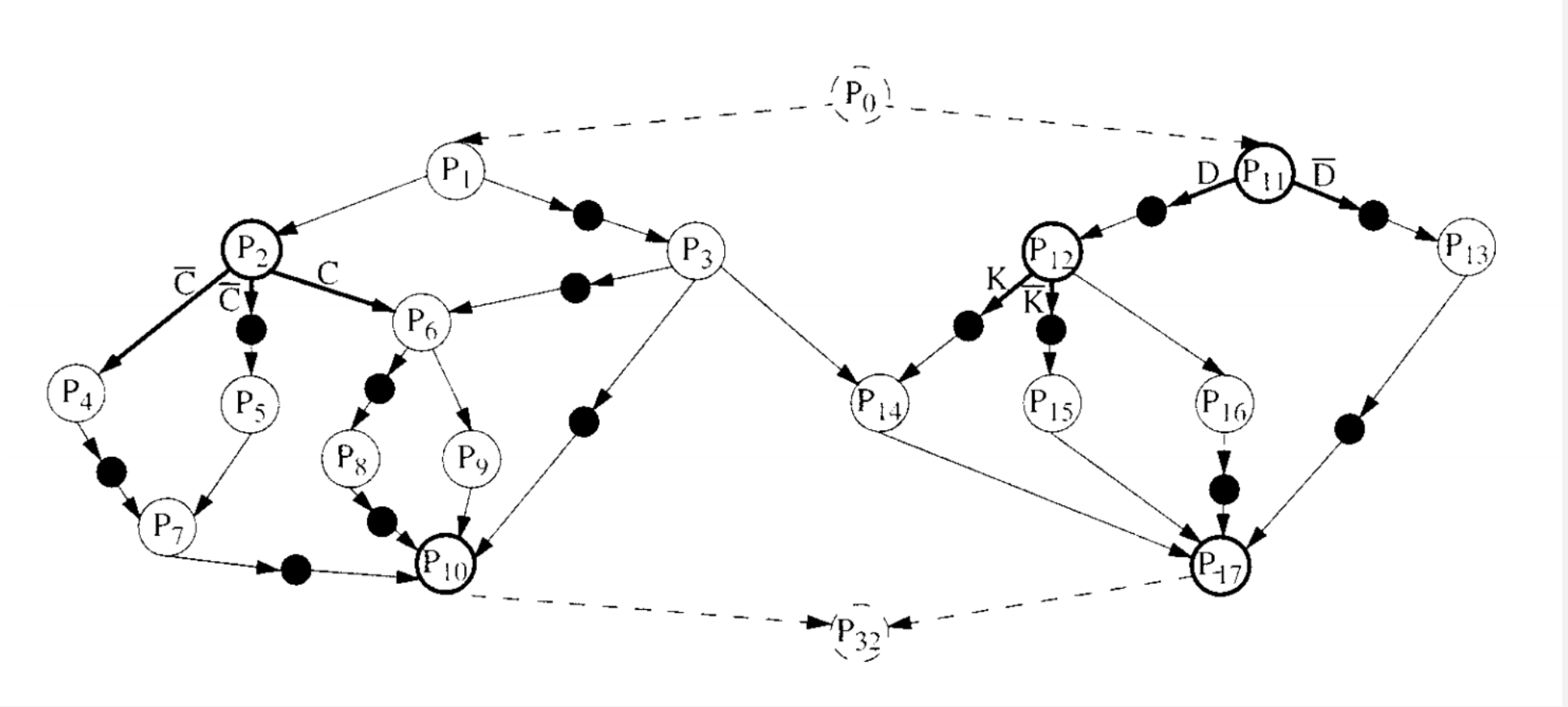
Alex Doboli
Linköping University

Paul Pop
Linköping University

Petru Eles
Linköping University

Zebo Peng
University of Cincinnati

# Heuristiken: CPG




[2]

Conditional Process Graph

G = (V, Es, Ec)

# Heuristiken: List Schedule

```
List_schedule ()
  for each processing element pe_i, i=1,2, ..., N_pe, do
    free_pei=0
  end for;
  schedule P_0 at t=0; initialize ready process list Ls_Ready
  with direct successors of P_0;        -- P_0 is the source node
  while Ls_Ready is not empty do
    p=Head(Ls_Ready);
    if M(p)∈ HP then                    -- hardware supports several
        schedule p at t=p.t_ready       -- processes at a time;
    else
        p=Select(Ls_Ready, M(p));
        schedule p at t=max(p.t_ready, free_M(p));
        free_M(p)=t+t_p
    end if;
    delete p from Ls_Ready;
    Ls_Ready = Ls_Ready + processes which become
                          ready after p is executed
  end while
end List_schedule;
```

[2]

Basic list scheduling algorithm

# Heuristiken: Schedule Table

| | *true* | $D$ | $D \wedge C$ | $D \wedge C \wedge \overline{K}$ | $D \wedge C \wedge K$ | $D \wedge \overline{C}$ | $D \wedge \overline{C} \wedge \overline{K}$ | $D \wedge \overline{C} \wedge K$ | $\overline{D}$ | $\overline{D} \wedge C$ | $\overline{D} \wedge \overline{C}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $P_1$ | 0 | | | | | | | | | | |
| $P_2$ | 3 | | | | | | | | | | |
| $P_3$ | | 6 | | | | | | | 6 | | |
| $P_4$ | | | | | | 7 | | | | | 7 |
| $P_5$ | | | | | | | 18 | 18 | | | 18 |
| $P_6$ | | | | 21 | 20 | | | | | 20 | |
| $P_7$ | | | | | | | 21 | 21 | | | 21 |
| $P_8$ | | | | 29 | 28 | | | | | 28 | |
| $P_9$ | | | | 26 | 25 | | | | | 25 | |
| $P_{10}$ | | | | 35 | 34 | | 27 | 26 | | 34 | 26 |
| $P_{11}$ | 0 | | | | | | | | | | |
| $P_{12}$ | | | 9 | | | 9 | | | | | |
| $P_{13}$ | | | | | | | | | | 10 | 13 |
| $P_{14}$ | | | | | 18 | | | 24 | | | |
| $P_{15}$ | | | | 19 | | | 24 | | | | |
| $P_{16}$ | | | | 15 | 15 | | 15 | 15 | | | |
| $P_{17}$ | | | | 25 | 24 | | 30 | 26 | | 24 | 24 |
| $P_{18}$ (1→3) | 3 | | | | | | | | | | |
| $P_{19}$ (2→5) | | | | | | 9 | | | | | 8 |
| $P_{20}$ (3→10) | | | | 21 | 20 | | 21 | 20 | | 20 | 20 |
| $P_{21}$ (3→6) | | | | 19 | 18 | | | | | 18 | |
| $P_{22}$ (4→7) | | | | | | 12 | | | | | 13 |
| $P_{23}$ (6→8) | | | | 26 | 25 | | | | | 25 | |
| $P_{24}$ (7→10) | | | | | | | 25 | 24 | | | 24 |
| $P_{25}$ (8→10) | | | | 33 | 32 | | | | | 32 | |
| $P_{26}$ (11→13) | | | | | | | | | | 8 | 11 |
| $P_{27}$ (11→12) | | | 8 | | | 8 | | | | | |
| $P_{28}$ (12→14) | | | | | 16 | | | 16 | | | |
| $P_{29}$ (12→15) | | | | 16 | | | 16 | | | | |
| $P_{30}$ (13→17) | | | | | | | | | | 22 | 22 |
| $P_{31}$ (16→17) | | | | 23 | 22 | | 23 | 22 | | | |
| D | 6 | | | | | | | | | | |
| C | | 7 | | | | | | | 7 | | |
| K | | | 15 | | | 15 | | | | | |

[2]

# Heuristiken: Ergebnis

- TTP ist kein TTE
- Keine Betrachtung von Nachrichtenklassen
- Kein reines statisches Scheduling
- Starker Fokus auf Conditions

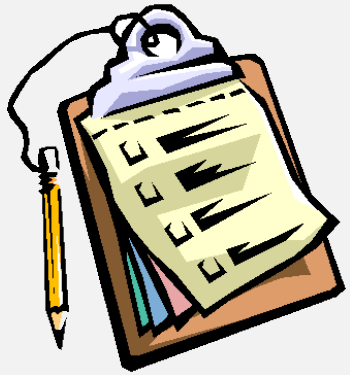
# Agenda

(1) Rückblick

**(2) Verwandte Arbeiten**

- Heuristiken
- **Model Checking (SPIN)**
- Model Checking (UPPAAL)

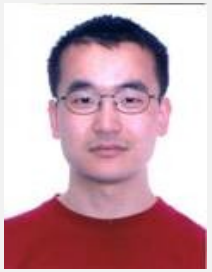
(3) Ausblick



# Model-Checking

## Optimization of Static Task and Bus Access Schedules for Time-Triggered Distributed Embedded Systems with Model-Checking

- Programmierbare Prozessoren
- TTP Bussystem
- Software Tasks und Network Messages in statischer Lookup Table
- Process Graph G = (V, E)



Zonghua Gu

Xiuqiang He

Mingxuan Yuan

Hong Kong University

# Model-Checking: SPIN

- Gerard J. Holzmann – Bell Labs
- Tool zum verifizieren von verteilten Software Modellen
- Promela (Process Meta Language)
- Linear Temporal Logic (LTL)
- Generiert Problem spezifische Model-Checker

# Model-Checking: Promela

```
proctype Task(byte i) {
{Block until precedence relations are satisfied.}
/*Block until CPU is free.*/
atomic{CPU[Task[i].cpuID].status==FREE->
  Task[i].status=RUNNING;
  Task[i].finTime=time+Task[i].et;
  CPU[Task[i].cpuID].status=BUSY;}
/*Block until time advances to its finish time*/
atomic{time==Task[i].finTime->
  Task[i].status=DONE;
  CPU[Task[i].cpuID].status=FREE;
  {Send messages to remote receiver tasks.}
```

[4]

Model of a non-preemptive Task

# Model-Checking: Ergebnis

- TTP ist kein TTE
- Liefert bessere Ergebnisse als die vorherige Heuristik
- Betrachtet work- und non work-conserving Scheduling
- Ressourcenintensiv
- State space explosion

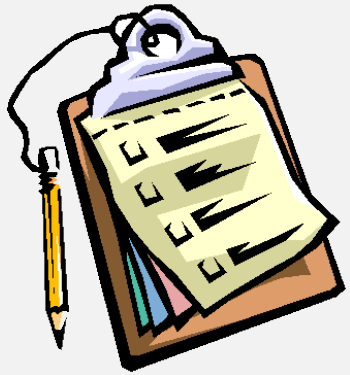
# Agenda

(1) Rückblick

**(2) Verwandte Arbeiten**

- Heuristiken
- Model Checking (SPIN)
- **Model Checking (UPPAAL)**

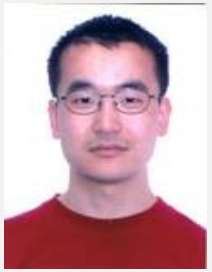
(3) Ausblick



# Model-Checking

## Optimal Static Task Scheduling on Reconfigurable Hardware Devices using Model-Checking

- Re-konfigurierbare FPGAs
- Kein verteiltes System
- Schedulingproblem ähnlich
- Parallelität
- UPPAAL



Zonghua Gu

Xiuqiang He

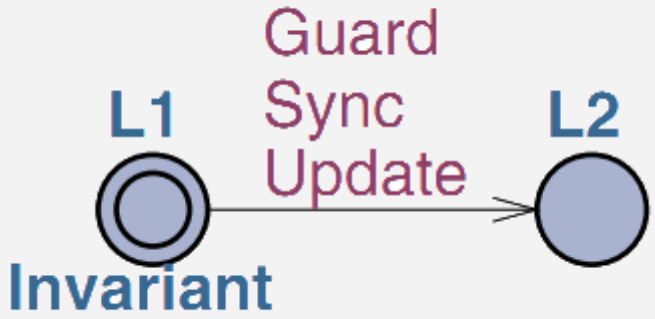
Mingxuan Yuan

Hong Kong University

# Model Checking: UPPAAL

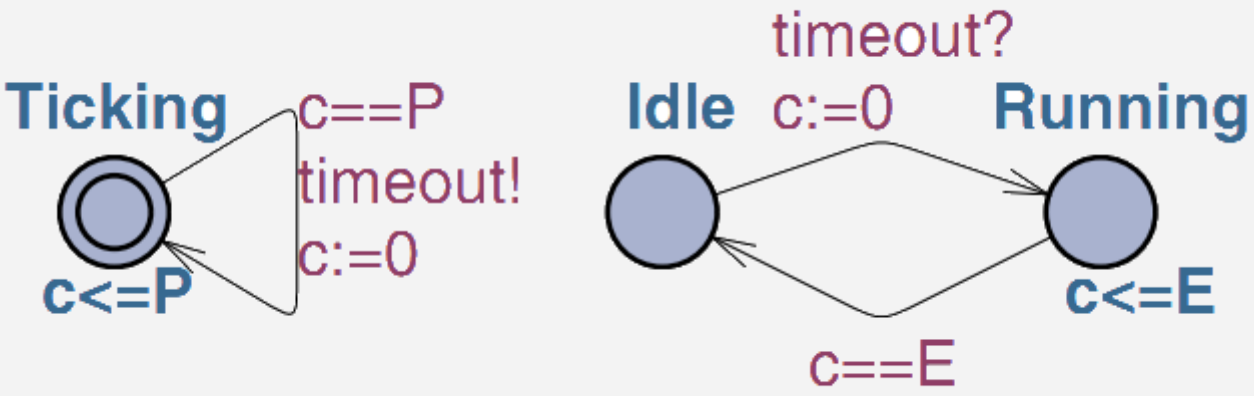
- Entwickelt von der Uppsala und Aalborg Universität
- Timed Automata
- Synchronisationskanäle
- Integer Variablen
- Timed Computational Tree Logic (TCTL)
- Execution Trace to final reachable state

# Model-Checking: UPPAAL




[5]

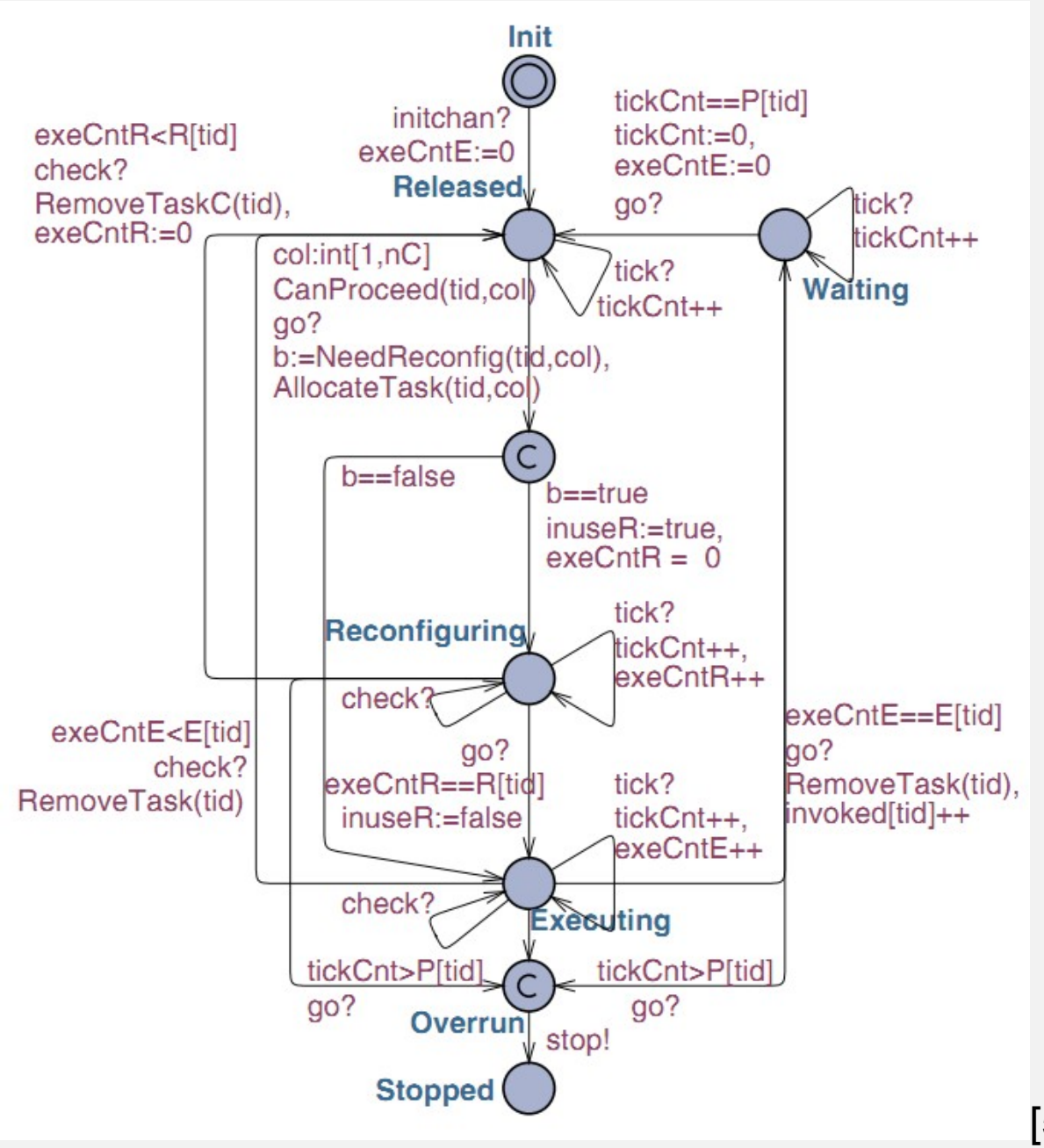
Basis Syntax




[5]

Periodic Task System

# Model-Checking: UPPAAL




[5]

# Model-Checking: Ergebnis

- Timed Automata alternativer Ansatz
- Static Scheduling of real-time tasks
- Parallelität
- Kein verteiltes System
- Kommunikationsprotokoll?
- Nachrichtenklassen?

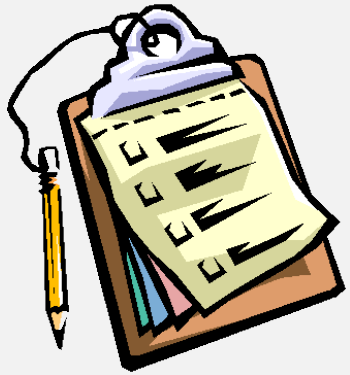
# Agenda

(1) Rückblick

(2) Verwandte Arbeiten

- Heuristiken
- Model Checking (SPIN)
- Model Checking (UPPAAL)

**(3) Ausblick**



# Ausblick: Projekt

- Basis-Tool entwickeln für TTE-Scheduling
- Heuristik oder Model-Checking?
- Wie lassen sich die TTE-Eigenschaften und Nachrichtenklassen integrieren?
- Bisherige Arbeiten beachten nur TTP

# Vielen Dank...

für die

Aufmerksamkeit!

# Referenzen


(1) Florian Bartols, Leistungsmessung von Time-Triggered Ethernet Komponenten unter harten Echtzeitbedingungen mithilfe modifizierter Linux-Treiber, 2010

(2) Petru Eles, Alex Doboli, Paul Pop and Zebo Peng, Scheduling with Bus Access Optimization for Distributed Embedded Systems, 2000

(3) Paul Pop, Petru Eles, Zebo Peng and Traian Pop, Scheduling and Mapping in an Incremental Design Methology for Distributed Real-Time Embedded Systems, 2004

(4) Zonghua Gu, Xiuqiang He and Mingxuan Yuan, Optimization of Static Task and Bus Access Schedules for Time-Triggered Distributed Embedded Systems width Model-Checking, 2007

(5) Gerard J. Holzmann, The Model Checker SPIN, 1997

(6) Zonghua Gu, Mingxuan Yuan and Xiuqiang He, Optimal Static Task Scheduling on Reconfigurable Hardware Devices using Model-Checking, 2007

(7) Fei Yu, Guoqiang Li and Naixue Xiong, Schedulability Analysis of Multi-Processor Real-Time Systems Using Uppaal, 2010