EDF Feasibility and Hardware Accelerators

Andrew Morton
University of Waterloo
Canada

Outline

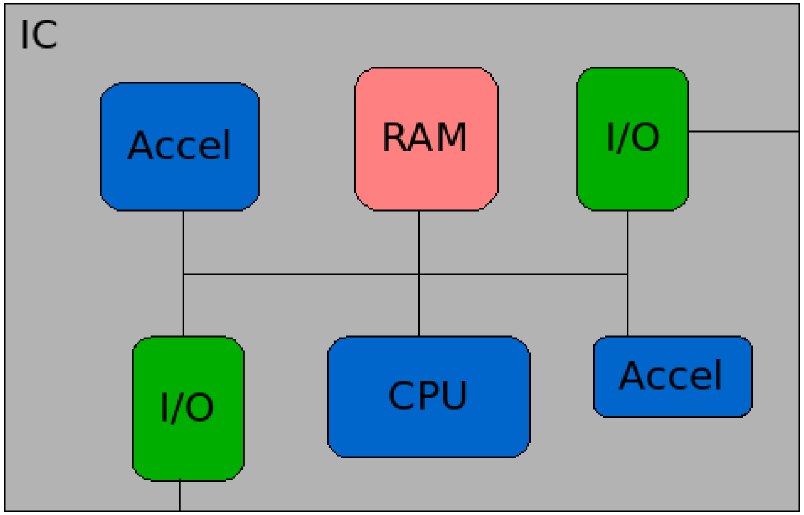
1) Introduction and motivation
2) Review of EDF and feasibility analysis
3) Hardware accelerators and scheduling
4) Extended analysis
5) Summary

Introduction

- Embedded systems
  - provides dedicated service to a larger system
  - digital embedded systems:
    - cell phone, washing machine, automotive, assembly-line
    - hard real-time is common
    - large volumes make production cost important

Introduction

- System-on-Chip
  - CPU(s), peripherals and custom logic on one IC
  - standard technique: use hardware accelerators to speed-up compute-intensive portion of software application



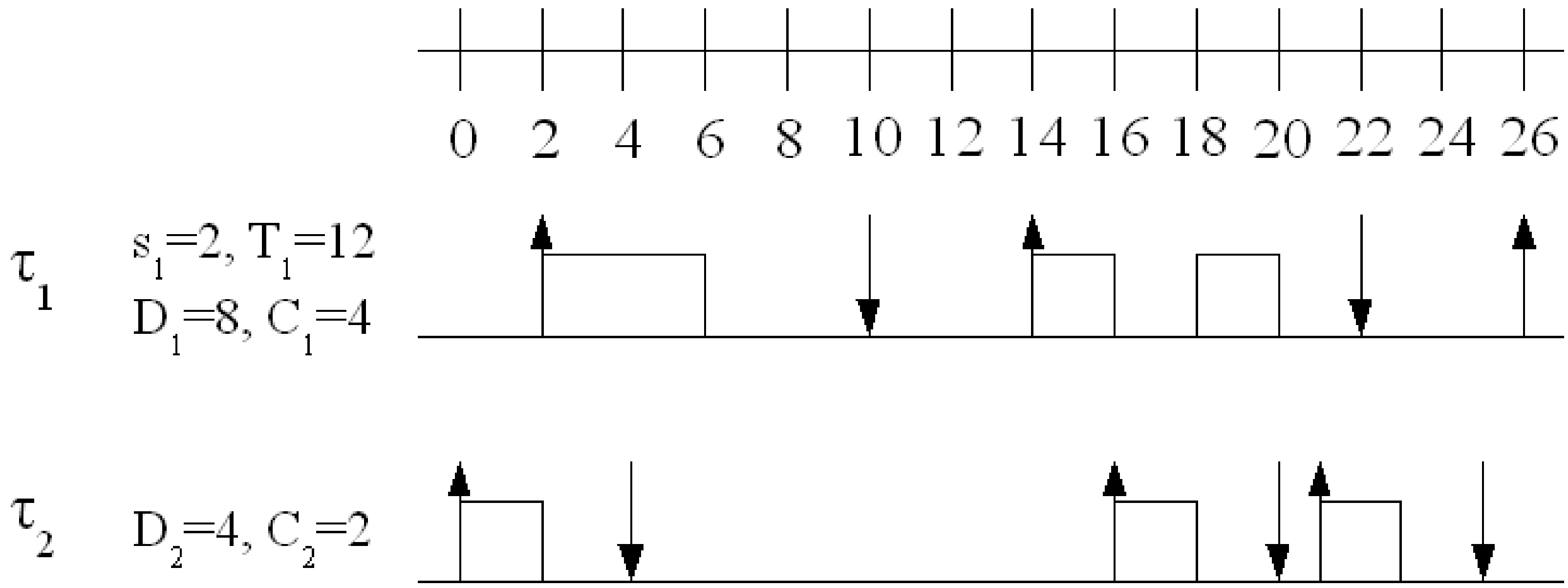
Introduction

- Typical Embedded Scheduling policies
  - table-based cyclic
  - fixed-priority preemptive
  - hybrid cyclic/preemptive
- Good alternative: Earliest Deadline First
  - EDF is optimal – will only miss a deadline if no other policy could make it
  - “natural” way to specify deadlines in embedded system

Earliest Deadline First (EDF)

- Definition of EDF
  - of all ready tasks, the task with the earliest deadline is executed first
  - if another task arrives with earlier deadline, it preempts the current task

Notation



- s_i start time, T_i period, C_i worst-case execution time, D_i deadline

Feasibility Analysis

- Given task set τ composed of periodic and aperiodic tasks:
  - determine whether all tasks can be scheduled by the preemptive EDF policy such that no task misses its deadline

Feasibility Analysis

- Technique
  - assume synchronous start
    - all tasks start at $t=0$
    - this is the “critical instance”
  - aperiodic task replaced by sporadic task with minimum inter-arrival time T_i
  - tools
    - processor utilization
    - processor demand

Processor Utilization: *U*

- Measure of overall processor business

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

  - sums the fraction of processor time required per task

Processor Demand: *h(t)*

- Measure of work required before time *t*

$$h(t) = \sum_{D_i \le t} \left(1 + \left\lfloor \frac{t - D_i}{T_i} \right\rfloor \right) C_i$$

  - sum execution times of all jobs (task instances) with deadlines not later than *t*

Feasibility Conditions

- Processor utilization does not exceed one

$$U \leq 1$$

- Processor demand never exceeds processor time

$$\forall t: \ h(t) \leq t$$

- Cannot check every *h(t)*

Feasibility Conditions

- Do not need to check every *h(t)*
  - processor demand only changes at job deadlines
    - only need to check at each deadline
  - can define upper limit on interval checked

Feasibility Conditions

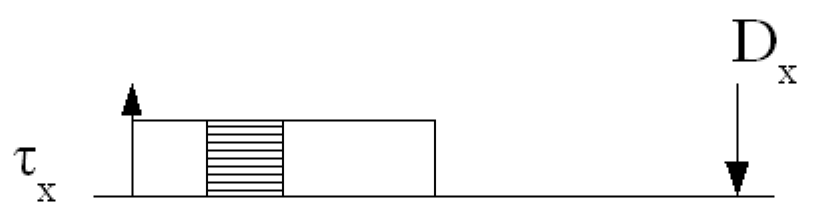
- upper limit for checking deadlines:
  - Theory [Liu and Layland, 1973] *When the deadline driven scheduling algorithm is used to schedule a set of tasks on a processor, there is no processor idle time prior to an overflow.*
  - interval before first idle called synchronous busy period (L)
  - only need to check until end of synchronous busy period

Feasibility Algorithm

```
if U>1 then
    return "infeasible"
endif
calculate L
for each deadline in [0,L]
    if h(t)>t
        return "infeasible"
    endif
endfor
return "feasible"
```

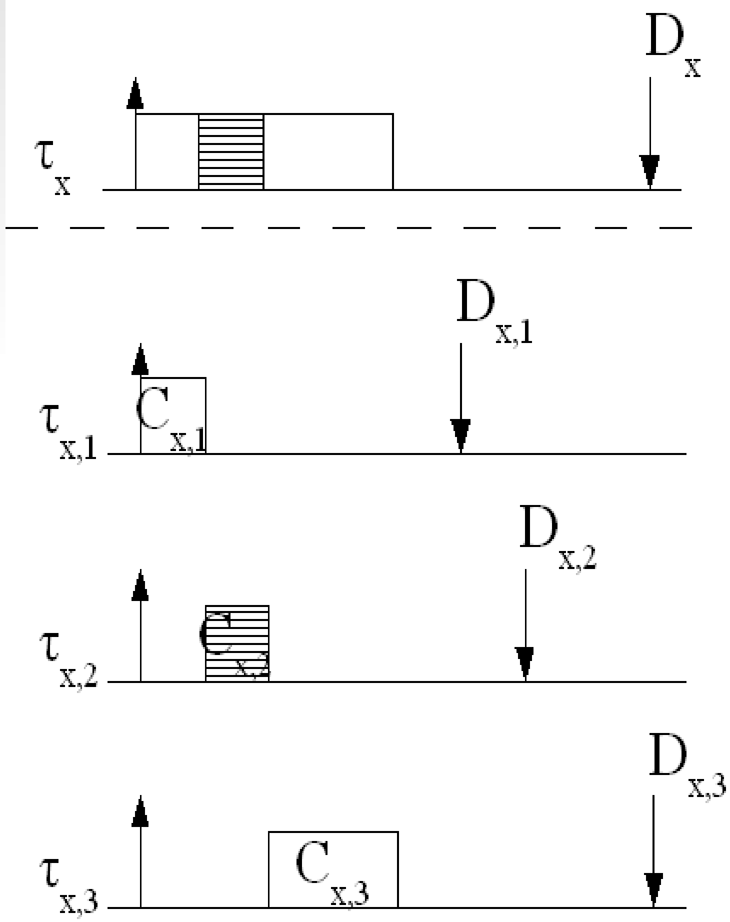
Accelerated Task

- Given an accelerator-blocked task



  - starts on CPU, transfers to accelerator, finishes on CPU
  - divide τ_x into subtasks $\tau_{x,1}$, $\tau_{x,2}$, and $\tau_{x,3}$

Accelerated Task Representation



$$D_{x,1} = D_{x,2} - C_{x,2}$$

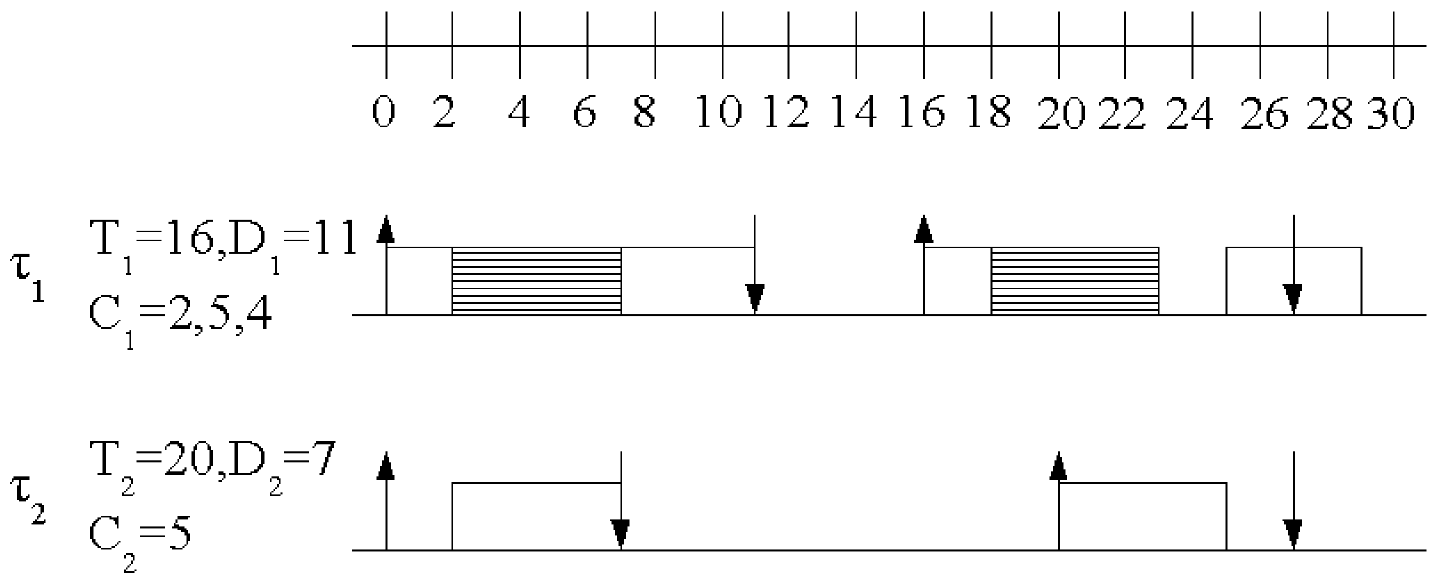
$$D_{x,2} = D_{x,3} - C_{x,3}$$

$$D_{x,3} = D_x$$

Extended Analysis – First Try

- Given task set τ with accelerator-blocked task τ_x:
  - replace τ_x with $\tau_{x,1}$ and $\tau_{x,3}$
  - perform feasibility analysis
- Problem: accelerator-induced idle periods

Extended Analysis - First Try

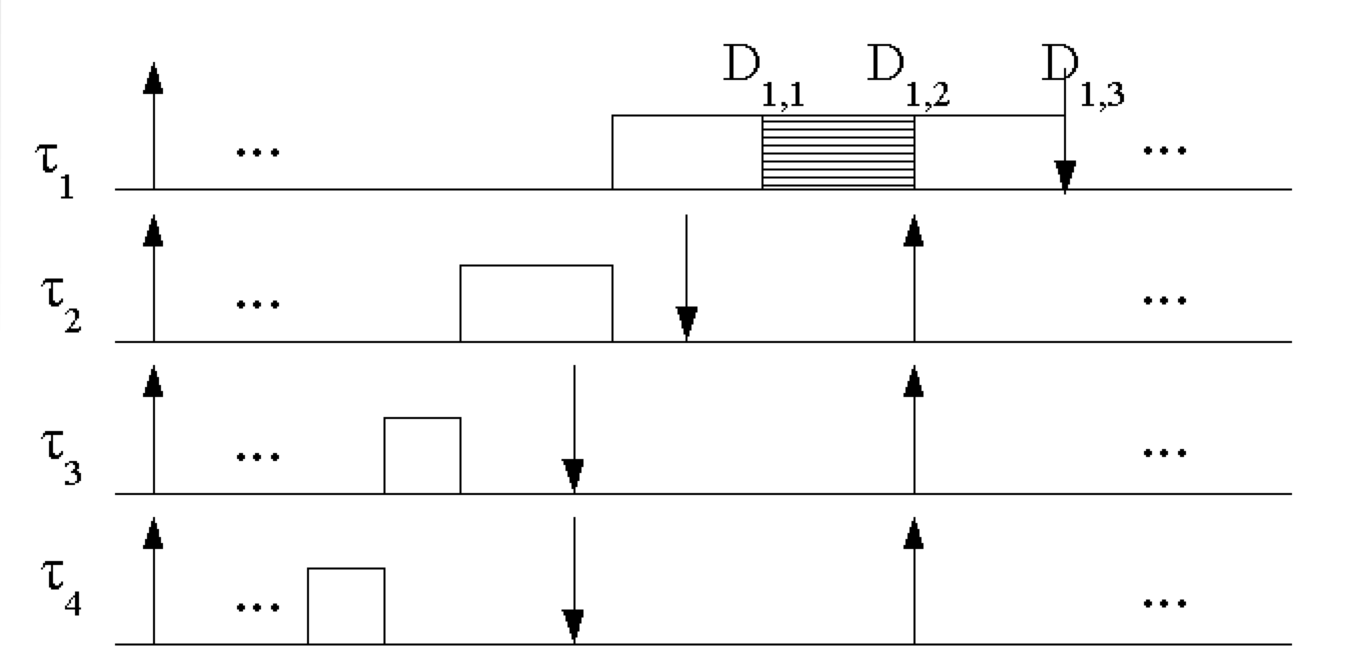


- this example would pass analysis
- missed dead-line at $t=27$
- during *[18,20]* CPU is idle because $\tau_{x,3}$ must wait for accelerator

Accelerator-Induced Idle

- Lemma1 *Given task set τ in which one task τ_x blocks on an accelerator once, with logical subtasks $\{\tau_{x,1}, \tau_{x,2}, \tau_{x,3}\}$, the accelerator-induced idle results in a critical instance when:*
    - *$\tau_{x,3}$ has no slack, and*
    - *all other tasks are released synchronously at the end of the accelerator-induced idle (i.e. at $t=D_{x,2}$).*

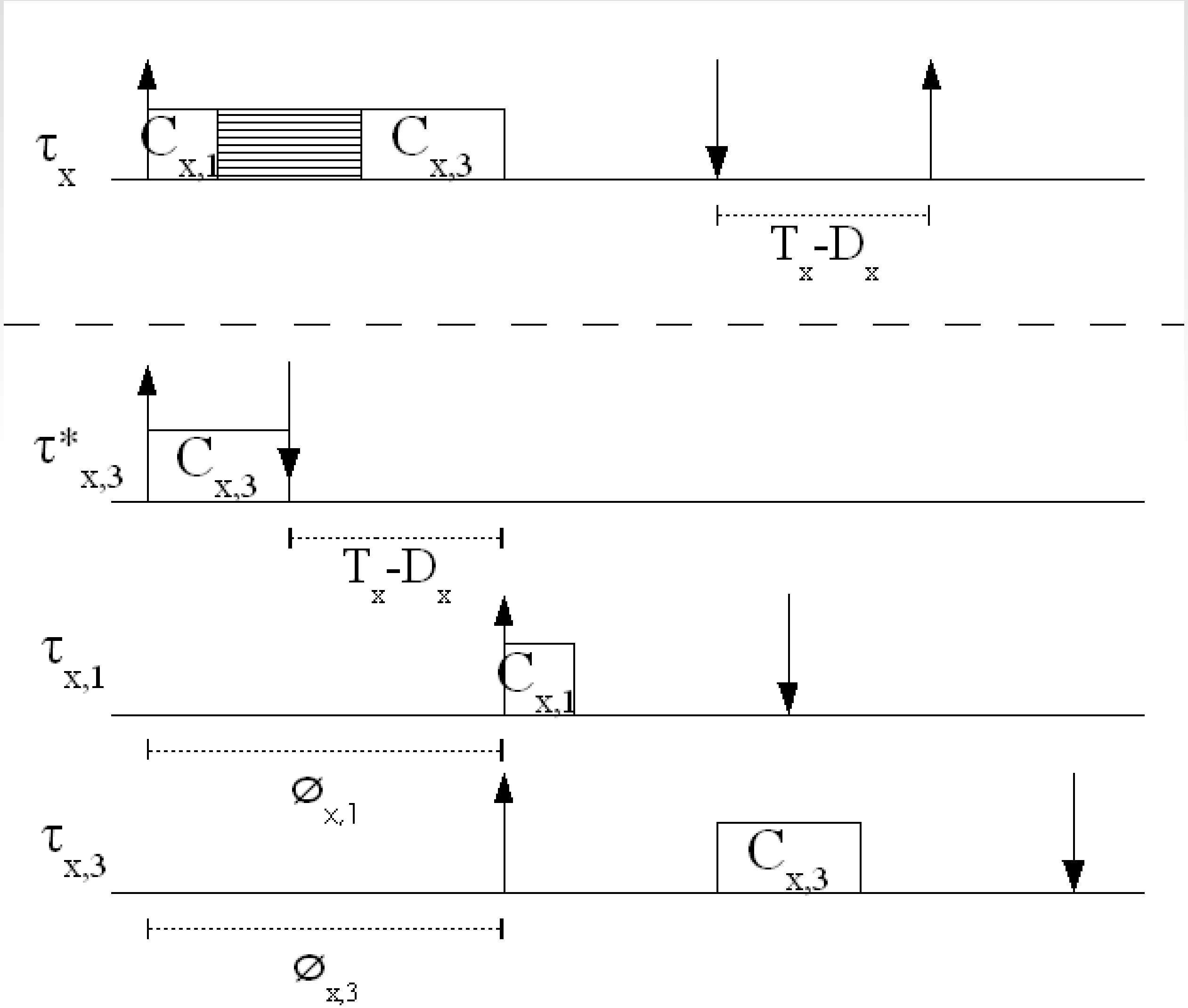
Accelerator-Induced Idle



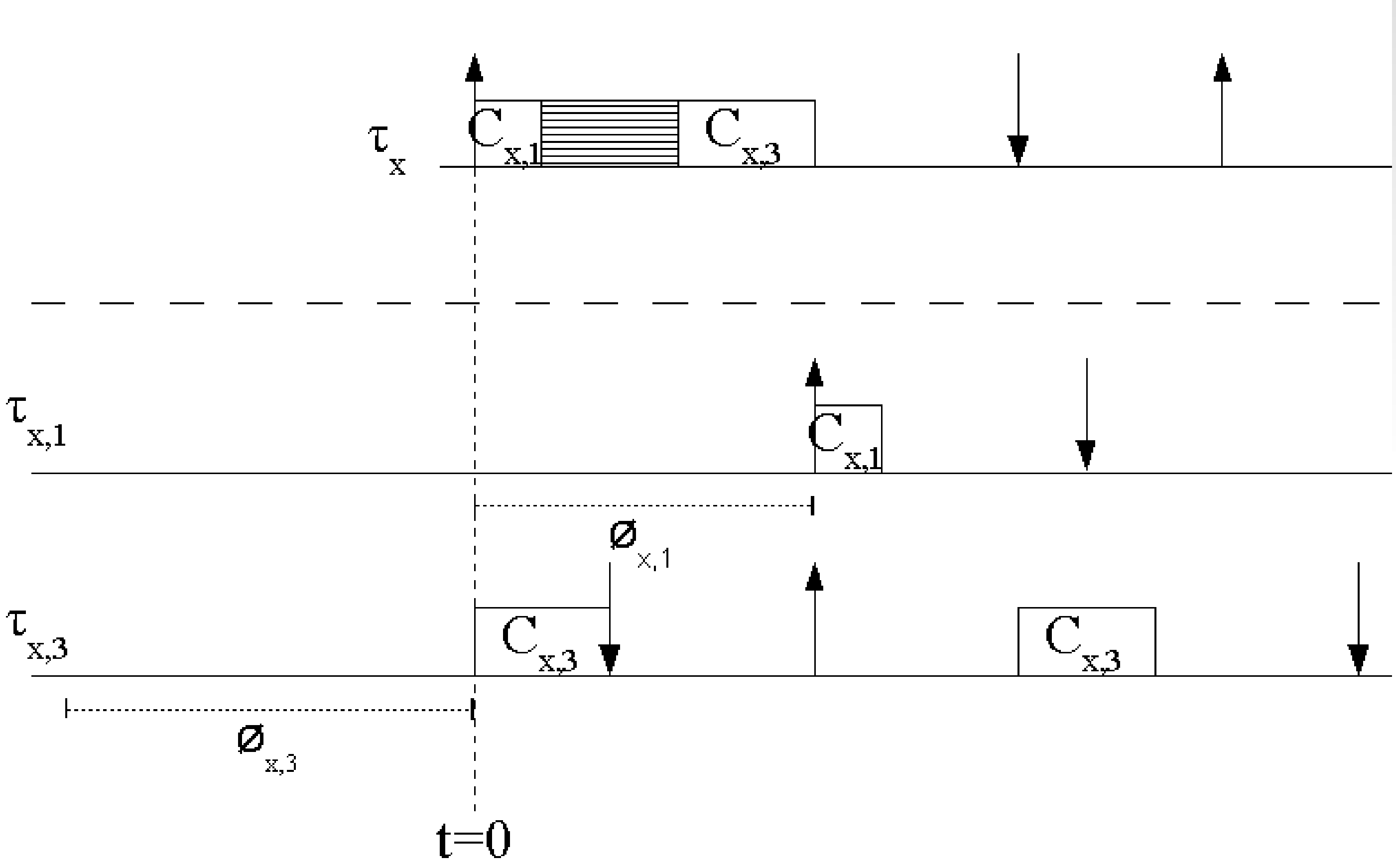
Extended Analysis

- Analysis of Accelerator-Induced Idle:
  - all regular tasks released at $t=0$
  - $\tau_{x,3}$ released at $t=0$
    - no slack → $D_{x,3}=C_{x,3}$
  - perform feasibility analysis

Subtask Phases



Subtask Phases



$$\phi_{x,1} = T_x - D_{x,2}$$

$$\phi_{x,3} = -D_{x,2}$$

Extended Algorithm

if $U>1$ then

  return "infeasible"

endif

calculate L

for each deadline in $[0,L]$

  if $h(t)>t$

    return "infeasible"

  endif

endfor

Pass 1

$\Phi_{x,1}=\Phi_{x,3}=0$

Pass 2

$\Phi_{x,1}=T_x-D_{x,2}$, $\Phi_{x,3}=-D_{x,2}$

modified h(t), L

$$h'(t)=\sum_{D_i+\phi_i\le t}\left(1+\left\lfloor\frac{t-(D_i+\phi_i)}{T_i}\right\rfloor\right)C_i$$

Summary

- Contributions
  - extended EDF feasibility analysis to task sets that include one accelerator-blocked task (that blocks once per task instance)
  - essentially heterogeneous multiprocessing with one preemtable and one non-preemptable processor
  - extends applicability of the EDF policy to embedded systems

Summary

- Future Work
  - extend to task sets that include one accelerator-blocked task that blocks multiple times (done)
  - extend to task sets that include multiple accelerator-blocked tasks