Hardware/Software Partitioning and Scheduling of Embedded Systems

Andrew Morton
PhD Thesis Defence
Electrical and Computer Engineering
University of Waterloo
January 13, 2005

Outline

1. Thesis Statement
2. Study 1: EDF with coprocessors
3. Study 2: SoPC case study
4. Study 3: Automated Partitioning
5. Summary

Thesis Statement

- Thesis: partitioning and scheduling of concurrent systems are closely connected and best approached in an integrated manner

Motivation

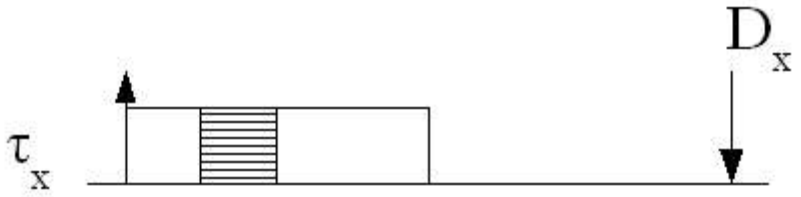
1. To explore relationship between partitioning and scheduling of embedded real-time systems
2. To apply Earliest Deadline First (EDF) scheduling policy in the context of hardware/software codesign
3. To integrate kernel partitioning with application partitioning

Dissertation Organization

- Study I
  - Extension of off-line EDF feasibility analysis for task sets that block on coprocessors
- Study II
  - Case study of application and kernel partitioning of a system on programmable chip (SoPC)
- Study III
  - Automated hardware/software partitioning of application and kernel for EDF feasibility

Study I: Extended EDF Analysis
Overview

- Problem
  - task set scheduled by EDF
    - task with earliest deadline is scheduled first; if another task arrives with earlier deadline, it preempts current task
  - a task blocks on coprocessor during execution



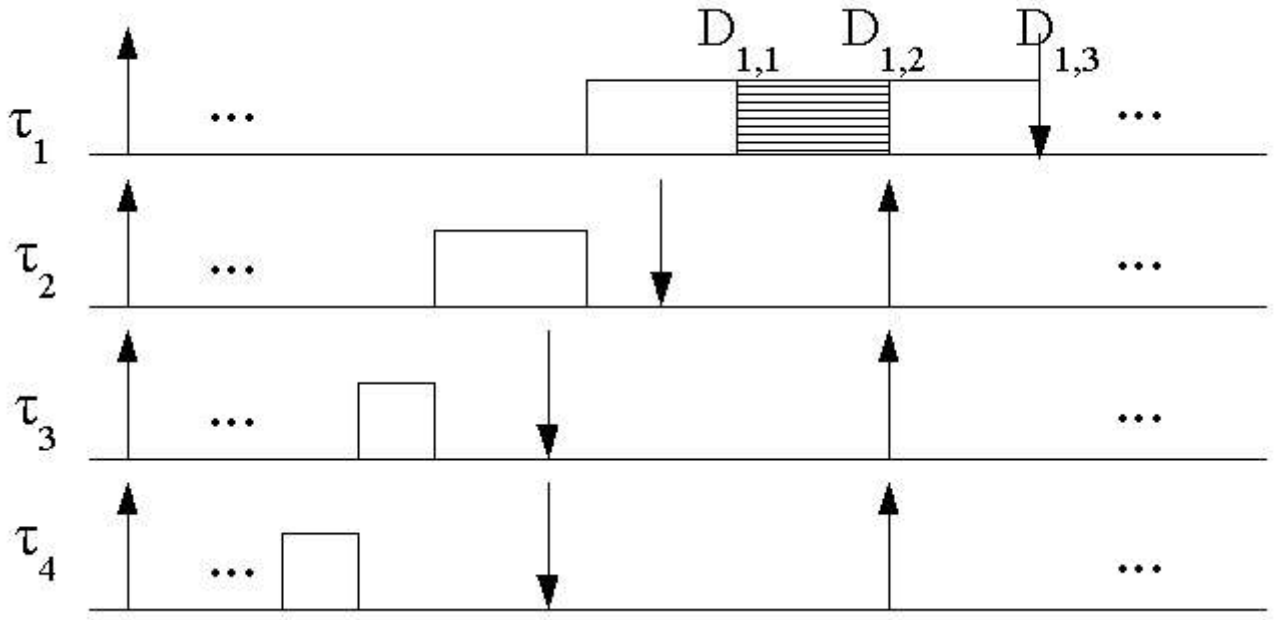
    - "coprocessor-blocked task"
  - highly constrained form of heterogeneous multiprocessor EDF

Extended EDF Analysis
Overview

- Analysis
  - Based on algorithm by Stankovic, Spuri, Ramamritham and Buttazzo (1998)
  - use processor demand analysis to ensure no missed deadlines
  - uni-processor, periodic and sporadic tasks
  - negligible kernel overhead
- Extended for task sets which include one coprocessor-blocked task
  - no contention for coprocessor

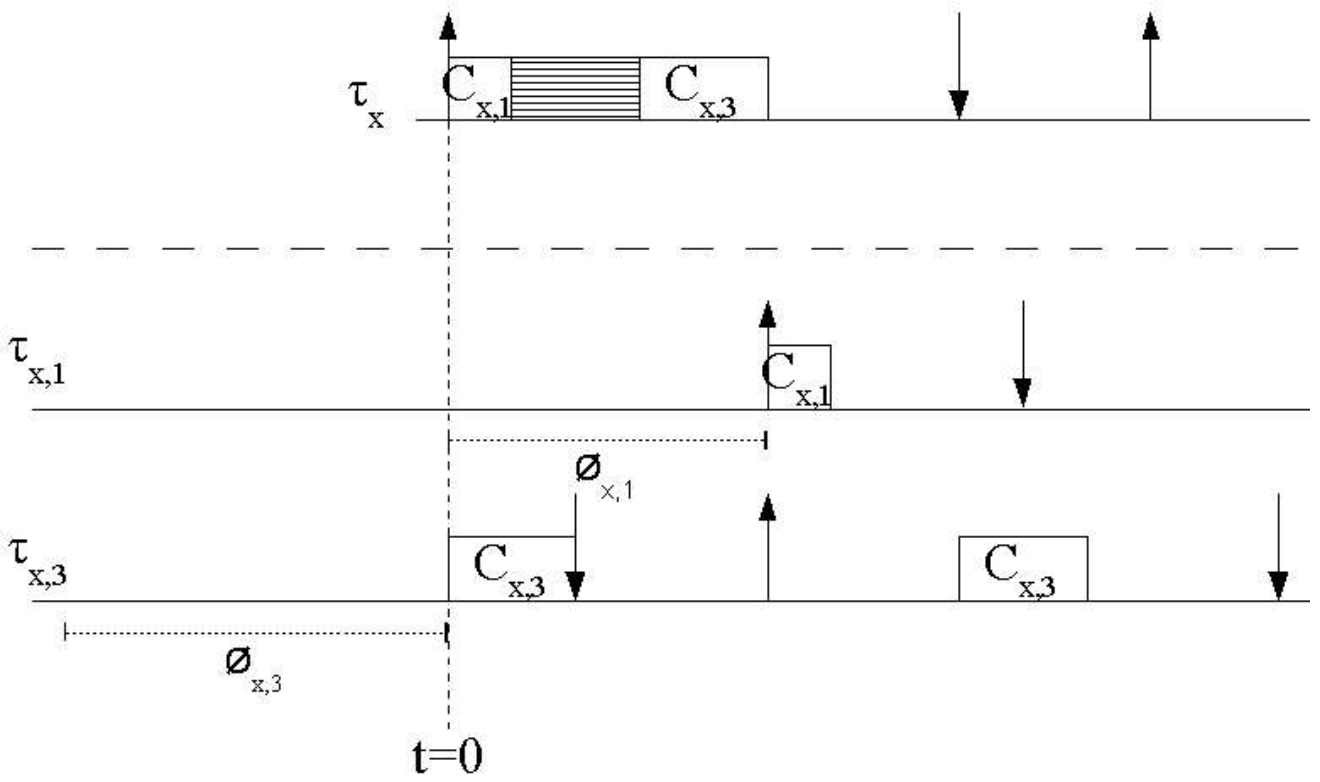
Extended EDF Analysis
Overview

- Analyze impact of coprocessor-induced idle



Extended EDF Analysis
Overview

- Assign phases to sub-tasks of coprocessor-blocked task



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

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

Extended EDF Analysis

Overview

- In τ replace τ_x with $\tau_{x,1}$, $\tau_{x,3}$ ... $\tau_{x,N}$
- Pass 1
  - $\Phi_{x,1}$, $\Phi_{x,3}$ ... $\Phi_{x,N} = 0$
  - perform processor demand analysis
- For each coprocessor block $\tau_{x,2}$, $\tau_{x,4}$ ... $\tau_{x,N-1}$
  - assign $\Phi_{x,1}$, $\Phi_{x,3}$ ... $\Phi_{x,N}$ to simulate coprocessor idle
  - perform processor demand analysis

Extended EDF Analysis
Overview

- Benefit

$$\Delta U = \frac{\Delta C_x}{T_x} = \frac{\sum_{k=2,4,\ldots,N-1} C_{x,k}}{T_x}$$

- Limitation

$$\forall |\vec{C}_i| = 1,$$
$$\forall k \in \{N-1, N-3, \ldots, 2\}: \quad D_i \geq C_i + C_{x,k+1}$$

Extended EDF Analysis
Contributions

1. Identified and characterized the problem of EDF feasibility analysis for coprocessor-blocked tasks
2. Proposed a first solution
3. Applied EDF in the context of hardware/software codesign
   - increases applicability of EDF to codesign

Extended EDF Analysis
Future Research

1. Develop analysis that doesn't impose limitation caused by subtask $\tau_{x,a}$ with zero slack
   - reconsider subtask deadline assignment
2. Extend analysis to multiple coprocessor-blocked tasks
3. Extend analysis to multiprocessor systems
4. Explore heuristics for dynamic planning of task sets with coprocessor-blocked tasks

Study II: SoPC Case Study
Overview

- System on Programmable Chip
  - FPGA with soft-core processor, system bus and coprocessors
- Real-time kernel
  - scheduling by EDF, inter-task message queues, integration of coprocessors
- Application
  - idle engine simulation, load simulation, crankshaft speed controllers

SoPC Case Study
Overview

- Application coprocessor
  - cosine calculation
  - used by environment task

	Cosine	Task
C++ cos()	1.435 ms	1.680 ms
cordic	0.081 ms	0.343 ms
difference	1.354 ms	1.337 ms*

* doesn't include kernel overhead

SoPC Case Study
Overview

- Kernel coprocessor
  - EDF scheduling
  - replaces run list, timer list and timer

	C_k
cs1	128 µs
cs2	66 µs
cs2 coproc	4.56 µs

SoPC Case Study
Overview

- Coprocessor Comparison

Coprocessor	Δt	ΔU	LE	$\Delta U/LE$ x 10^6
cordic	1.081 ms	0.2162	3840	56.30
cs2	62 µs	0.1036	1836	56.42

SoPC Case Study
Contributions

1. One more hardware/software partitioning of kernel
2. Proposed coprocessor evaluation metric
   - $\Delta U/LE$
3. Real-time kernel with integrated coprocessor support
4. Data source for automated hardware/software partitioners

SoPC Case Study
Future Research

1. Perform additional case-studies of real-time SoC systems
2. Implement kernel Slif nodes on FPGA to check hardware estimates and schedule feasibility
3. Compare multi-processor implementations against processor/coprocessor implementations

Study III: Hardware/Software Partitioning
Overview

- Problem
  - Given one processor with fixed throughput, limited programmable logic, and possibly limited program/data memory, partition the application and kernel such that all deadlines are met when scheduled by the preemptive EDF policy.

Hardware/Software Partitioning
Overview

- Non-linear programming (NLP) model
- Objective: EDF feasibility
  1. minimize U
  2. repair schedule feasibility
     - add processor demand constraints

Hardware/Software Partitioning
Overview

$$U = \sum_{\tau_i \in \tau} \frac{C_i}{T_i}$$

$$C_i = \sum_{n_a \in N_i} \left(c_{i,a}[hw] \cdot n_a[hw] + c_{i,a}[sw] \cdot n_a[sw] \right)$$
$$+ 2\, C_k \left(n_{root}[sw] + \sum_{e_p \in E_i} f_p \left(e_p[h/s] + e_p[s/h] \right) \right)$$

$$C_k = \sum_{n_a \in N_k} \left(c_{k,a}[hw] \cdot n_a[hw] + c_{k,a}[sw] \cdot n_a[sw] \right)$$

Hardware/Software Partitioning
Overview

- FM-based heuristic
- Objective: EDF feasibility

1. minimize

   a) worst-case execution time – metric g^c
   b) scaled, weighted cutset – metric g^χ
   c) processor utilization – metric g^U
   d) processor utilization divided by size – metric g^U/sz

2. repair schedule feasibility by

   - minimize $h(v_k)$ for first missed deadline
   - repeat until no missed deadlines

Hardware/Software Partitioning
Overview

- Idle Engine Partitioning
  - kernel nodes: 11 (5 bound to software)
  - task nodes: 30
  - U (all software) = 1.097
- NLP results
  - U = 0.465680
  - hardware nodes
    - kernel: 5
    - task: 23
- Case study results (cordic)
  - U = 0.881101
  - hardware nodes
    - kernel: 0
    - task: 1

Hardware/Software Partitioning
Overview

- Heuristic gain metric evaluation results (part 1)

Problem	# Feasible			
	g^c	g^χ	g^U	$g^{U/sz}$
idle eng	0	0	31	100
p100_1	0	40	82	83
p100_2	0	3	100	100
p200_1	92	0	92	98
p200_2	0	3	100	100
p500_1	0	0	4	0
p500_2	0	5	100	99
p1000_1	0	14	68	52
p1000_2	0	1	100	100
Total	92	66	677	732

Hardware/Software Partitioning
Overview

- Heuristic gain metric evaluation results (part 2)

	g^{U}		$g^{U/sz}$	
Problem	*U* min	*U* std dev	*U* min	*U* std dev
idle eng	0.4657	0.01881	0.4657	0.01350
p100_1	0.8984	0	0.8984	0
p100_2	0.8512	0	0.8512	0
p200_1	0.6772	0.02347	0.6772	0.02316
p200_2	0.7836	0	0.7836	0
p500_1	0.5538	0.006246	---	---
p500_2	0.7064	0	0.7064	0
p1000_1	0.7739	0.04015	0.8275	0.02619
p1000_2	0.2445	0	0.2445	0

Hardware/Software Partitioning
Overview

- Kernel vs Application hardware assignment

Problem	Kernel	Task
idle eng	45.4%	76.7%
p100_1	36.4%	3.0%
p100_2	36.4%	1.0%
p200_1	9.1%	2.5%
p200_2	45.4%	0.0%
p500_1	45.4%	6.2%
p500_2	63.6%	0.0%
p1000_1	36.4%	3.3%
p1000_2	54.5%	0.2%

Hardware/Software Partitioning
Contributions

1. Addressed schedule feasibility during partitioning
2. Minimizing U helps find feasible schedules
3. Partitioned kernel nodes contribute to feasible schedules
4. Design time should be a constraint
   - encourage code re-use (including kernel)
5. Results indicate that schedule feasibility needs to be addressed during partitioning

Hardware/Software Partitioning
Future Research

1. Use finer-grained internal representation such as control and data-flow graphs
2. Extend to multiprocessor partitioning
3. Synthesis of hardware coprocessors from software code would make automated partitioning a more applicable technique

Summary

- Demonstrated importance of addressing schedule feasibility during partitioning
- Explored EDF in context of hardware/software codesign
- Results indicate that the kernel is a viable candidate for partitioning because of invocation frequency and code re-use