Tight Schedule: Deadline Constrained Scheduling of OpenRadio on a Multi-Core Platform

C. Copeland  Y. Michalevsky  S. Michelson

Department of Electrical Engineering
Stanford University

CS316: Advanced Multi-Core Architecture, 2012
Outline

1 Motivation
   - OpenRadio platform [1]
   - Requirements

2 Scheduling Methods
   - Static Scheduling
   - Dynamic Scheduling

3 Experimental Setup and Results
Outline

1 Motivation
   • OpenRadio platform[1]
   • Requirements

2 Scheduling Methods
   • Static Scheduling
   • Dynamic Scheduling

3 Experimental Setup and Results
OpenRadio platform

- Wireless baseband processing
- Programmable PHY
- Affordable hardware
  - TI KeyStone Platform
- Heterogeneous multi-core platform
  - TI DSP C64 × 3
  - Viterbi Co-processor × 4
  - FFT Co-processor × 2

Figure: TI KeyStone platform
Outline

1 Motivation
   - OpenRadio platform[1]
   - Requirements

2 Scheduling Methods
   - Static Scheduling
   - Dynamic Scheduling

3 Experimental Setup and Results
Scheduling Requirements

**Imposed by the protocol**
- Timeout for sending ACK - 15µs since last symbol has been TX-ed.
- Number of OFDM symbols per packet can vary.

**System Requirements**
- Enable frequent modifications
  - Compile-time (reasonable computation time)
  - Runtime
- Low footprint (in case of runtime scheduling)
Scheduling Requirements

Imposed by the protocol

- Timeout for sending ACK - 15\(\mu\)s since last symbol has been TX-ed.
- Number of OFDM symbols per packet can vary.

System Requirements

- Enable frequent modifications
  - Compile-time (reasonable computation time)
  - Runtime
- Low footprint (in case of runtime scheduling)
Scheduling Requirements

**Imposed by the protocol**
- Timeout for sending ACK - 15\(\mu s\) since last symbol has been TX-ed.
- Number of OFDM symbols per packet can vary.

**System Requirements**
- Enable frequent modifications
  - Compile-time (reasonable computation time)
  - Runtime
- Low footprint (in case of runtime scheduling)
Scheduling Methods

- **Static**
  - Formal – optimization problem
    - Optimal solution (according to some objective)
  - Heuristic
    - Suboptimal

- **Dynamic**
  - Simulation yields a schedule that can be used as static
Scheduling Methods

- **Static**
  - Formal – optimization problem
    - Optimal solution (according to some objective)
  - Heuristic
    - Suboptimal

- **Dynamic**
  - Simulation yields a schedule that can be used as static
Scheduling Methods

- **Static**
  - Formal – optimization problem
    - Optimal solution (according to some objective)
  - Heuristic
    - Suboptimal

- **Dynamic**
  - Simulation yields a schedule that can be used as static
Scheduling Methods

- **Static**
  - Formal – optimization problem
    - Optimal solution (according to some objective)
  - Heuristic
    - Suboptimal

- **Dynamic**
  - Simulation yields a schedule that can be used as static
Scheduling Methods

- **Static**
  - Formal – optimization problem
    * Optimal solution (according to some objective)
  - Heuristic
    * Suboptimal

- **Dynamic**
  - Simulation yields a schedule that can be used as static
Outline

1 Motivation
   • OpenRadio platform[1]
   • Requirements

2 Scheduling Methods
   • Static Scheduling
   • Dynamic Scheduling

3 Experimental Setup and Results
Static Scheduling

- Previous work in SNSG resulted in a mixed integer linear programming model (Mixed-ILP) of OpenRadio task schedules (Based on scheduling problem formulation techniques such as [2, 3])
- Solving the model produces optimal schedules
- We can also produce all feasible schedules
- Different operating conditions for more complex workflows may change which is optimal (Wi-Fi is deterministic)
- Using IBM CPLEX solver, we can now find optimal schedules for 54 Mbps Wi-Fi
Static Scheduling

Constraints

- Each task is assigned to exactly one processor
- Tasks must be scheduled after their dependencies (plus communication delay if using a different core)
- No overlapping of tasks on the same processor
- The schedule must be "tilable" – multiple copies of the tasks must be scheduled on the same hardware using a period that is smaller than the deadline

Each of these high level constraints corresponds to a set of equalities and inequalities in a Mixed-ILP.

Generated automatically from the flow-graph for a particular wireless scenario.
Static Scheduling

Constraints

- Each task is assigned to exactly one processor
- Tasks must be scheduled after their dependencies (plus communication delay if using a different core)
- No overlapping of tasks on the same processor
- The schedule must be "tilable" – multiple copies of the tasks must be scheduled on the same hardware using a period that is smaller than the deadline

- Each of these high level constraints corresponds to a set of equalities and inequalities in a Mixed-ILP.
- Generated automatically from the flow-graph for a particular wireless scenario.
Static Scheduling

Constraints

- Each task is assigned to exactly one processor
- Tasks must be scheduled after their dependencies (plus communication delay if using a different core)
- No overlapping of tasks on the same processor
- The schedule must be "tilable" – multiple copies of the tasks must be scheduled on the same hardware using a period that is smaller than the deadline

- Each of these high level constraints corresponds to a set of equalities and inequalities in a Mixed-ILP.
- Generated automatically from the flow-graph for a particular wireless scenario.
Static Scheduling

Constraints

- Each task is assigned to exactly one processor
- Tasks must be scheduled after their dependencies (plus communication delay if using a different core)
- No overlapping of tasks on the same processor
- The schedule must be "tilable" – multiple copies of the tasks must be scheduled on the same hardware using a period that is smaller than the deadline

Each of these high level constraints corresponds to a set of equalities and inequalities in a Mixed-ILP.

Generated automatically from the flow-graph for a particular wireless scenario.
Static Scheduling

Constraints

- Each task is assigned to exactly one processor
- Tasks must be scheduled after their dependencies (plus communication delay if using a different core)
- No overlapping of tasks on the same processor
- The schedule must be "tilable" – multiple copies of the tasks must be scheduled on the same hardware using a period that is smaller than the deadline

- Each of these high level constraints corresponds to a set of equalities and inequalities in a Mixed-ILP.
- Generated automatically from the flow-graph for a particular wireless scenario.
Static Scheduling
Constraints

- Each task is assigned to exactly one processor
- Tasks must be scheduled after their dependencies (plus communication delay if using a different core)
- No overlapping of tasks on the same processor
- The schedule must be "tilable" – multiple copies of the tasks must be scheduled on the same hardware using a period that is smaller than the deadline

- Each of these high level constraints corresponds to a set of equalities and inequalities in a Mixed-ILP.
- Generated automatically from the flow-graph for a particular wireless scenario.
Static Scheduling
Potential Tweaks and Extension

What is optimal?
Change objective function to encapsulate parameters other than deadline slack:
  ▶ Hardware utilization (maybe the same job can be done with less)
  ▶ Power consumption
  ▶ Provide the solver with unlimited resources (processors) of different kinds and let it find the "best" hardware configuration and schedule for your needs
What is optimal?

Change objective function to encapsulate parameters other than deadline slack:
  - Hardware utilization (maybe the same job can be done with less)
  - Power consumption
  - Provide the solver with unlimited resources (processors) of different kinds and let it find the "best" hardware configuration and schedule for your needs
Static Scheduling
Potential Tweaks and Extension

- What is optimal?
- Change objective function to encapsulate parameters other than deadline slack:
  - Hardware utilization (maybe the same job can be done with less)
  - Power consumption
  - Provide the solver with unlimited resources (processors) of different kinds and let it find the "best" hardware configuration and schedule for your needs
Static Scheduling
Potential Tweaks and Extension

- What is optimal?
- Change objective function to encapsulate parameters other than deadline slack:
  - Hardware utilization (maybe the same job can be done with less)
  - Power consumption
  - Provide the solver with unlimited resources (processors) of different kinds and let it find the "best" hardware configuration and schedule for your needs
Outline

1 Motivation
   • OpenRadio platform[1]
   • Requirements

2 Scheduling Methods
   • Static Scheduling
   • Dynamic Scheduling

3 Experimental Setup and Results
Dynamic Scheduling
Forward Surplus

Figure: An example of computing FSN, FST, and FSR for arriving task T6 (not yet assigned to a resource) in a system with three resources and five previously-scheduled tasks [5].
Dynamic Scheduling

Deadline Setting

Figure: Critical path scheduling example

- Deadlines currently determined by critical path
- Can be decided and altered offline
- Many potential tweaks to scheduling
  - None affect overhead
Dynamic Scheduling

Deadline Setting

- Deadlines currently determined by critical path
- Can be decided and altered offline
- Many potential tweaks to scheduling
  - None affect overhead

Figure: Critical path scheduling example
Dynamic Scheduling

Deadline Setting

- Deadlines currently determined by critical path
- Can be decided and altered offline
- Many potential tweaks to scheduling
  - None affect overhead

Figure: Critical path scheduling example
Dynamic Scheduling

Deadline Setting

- Deadlines currently determined by critical path
- Can be decided and altered offline
- Many potential tweaks to scheduling
  - None affect overhead

Figure: Critical path scheduling example
Experiments and Results

54 Mbps workflow

Feasible schedules for toughest Wi-Fi rate

Figure: Scheduling 54 Mbps workflow using Dynamic and Static approaches. Static schedule repeated for 4 symbols yields a slack of 818 cycles. Dynamic schedule repeated for 4 symbols yields a slack of 218 cycles. (Visualization using Torsche Scheduling Toolbox [4])
Experiments and Results

Dependency on interprocessor communication latency

Figure: Dependency of schedules on interprocessor communication latency

Difference of 400 cycles between maximum and minimum makespan. Communication times do not seem to play a crucial role.
Experiments and Results
Possibly feasible to handle 54 Mbps with 2 DSPs only

Figure: We tried to find a feasible schedule using only 2 DSPs. Successfully. 618 Slack cycles.
Summary

● Possible to meet Wi-Fi deadlines on given hardware platform.
● Dynamic scheduling yields comparable results to static scheduling and might be useful even for hard-real time systems.
● Study of platform parameters influence.

Outlook

▶ To test schedules on OpenRadio platform.
▶ Application of constraints minimization techniques for Mixed-ILP solution.
▶ Dynamic scheduling
  ★ Heuristic optimizations: Speculative shortcut decisions, Task grouping
  ★ Implement and test with OpenRadio platform
**Summary**

- Possible to meet Wi-Fi deadlines on given hardware platform.
- Dynamic scheduling yields comparable results to static scheduling and might be useful even for hard-real time systems.
- Study of platform parameters influence.

**Outlook**

- To test schedules on OpenRadio platform.
- Application of constraints minimization techniques for Mixed-ILP solution.
- Dynamic scheduling
  - Heuristic optimizations: Speculative shortcut decisions, Task grouping
  - Implement and test with OpenRadio platform
Repeatable Schedule

Example

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Repeatable Schedule

Example

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
802.11a RX Chain

Figure: RX Chain
References


Cyclic scheduling of tasks with unit processing time on dedicated sets of parallel identical processors.

Deadline constrained cyclic scheduling on pipelined dedicated processors considering multiprocessor tasks and changeover times.

Torsche scheduling toolbox for matlab.

Dynamic binding and scheduling of firm-deadline tasks on heterogeneous compute resources.